Ma 115

ACTA
POLYTECHNICA
SCANDINAVICA

MATHEMATICS AND COMPUTING SERIES No. 115

Data Exploration Process Based on the Self−Organizing Map

JUHA VESANTO

Helsinki University of Technology
Department of Computer Science and Engineering
FIN−02015 HUT
FINLAND

Dissertation for the degree of Doctor of Technology to be presented with due permission of Computer Science and Engineering for public examination and debate at Helsinki University of Technology, Espoo, Finland on the 16th of May, 2002, at 12 o’clock noon.

ESPOO 2002

Vesanto, J., **Data Exploration Process Based on the Self−Organizing Map**, Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 115, Espoo 2002, 96 pp. Published by the Finnish Academies of Technology. ISBN 951−666−596−9, ISSN 1456−9418.

Keywords: self−organizing map, exploratory data analysis, data mining, visualization, clustering, vector quantization.

ABSTRACT

With the advances in computer technology, the amount of data that is obtained from various sources and stored in electronic media is growing at exponential rates. Data mining is a research area which answers to the challange of analysing this data in order to find useful information contained therein. The Self−Organizing Map (SOM) is one of the methods used in data mining. It quantizes the training data into a representative set of prototype vectors and maps them on a low−dimensional grid. The SOM is a prominent tool in the initial exploratory phase in data mining.

The thesis consists of an introduction and ten publications. In the publications, the validity of SOM−based data exploration methods has been investigated and various enhancements to them have been proposed. In the introduction, these methods are presented as parts of the data mining process, and they are compared with other data exploration methods with similar aims.

The work makes two primary contributions. Firstly, it has been shown that the SOM provides a versatile platform on top of which various data exploration methods can be efficiently constructed. New methods and measures for visualization of data, clustering, cluster characterization, and quantization have been proposed. The SOM algorithm and the proposed methods and measures have been implemented as a set of Matlab routines in the SOM Toolbox software library.

Secondly, a framework for SOM−based data exploration of table−format data − both single tables and hierarchically organized tables − has been constructed. The framework divides exploratory data analysis into several sub−tasks, most notably the analysis of samples and the analysis of variables. The analysis methods are applied autonomously and their results are provided in a report describing the most important properties of the data manifold. In such a framework, the attention of the data miner can be directed more towards the actual data exploration task, rather than on the application of the analysis methods. Because of the highly iterative nature of the data exploration, the automation of routine analysis tasks can reduce the time needed by the data exploration process considerably.

Preface

This doctoral thesis has been made in the Laboratory of Computer and Information Science in the Helsinki University of Technology, in 1997–2002.

I most sincerely thank Professor Olli Simula for his supervision and encouragement, as well as him and Academy Professor Erkki Oja for their support during this work. The Laboratory of Computer and Information Science together with the Neural Networks Research Centre has provided an excellent environment for research, both in terms of facilities and inspiration. I also feel much obliged to Academician Teuvo Kohonen on whose work this thesis, as well as the existence of the Neural Networks Research Centre, is ultimately based on.

I am deeply indebted to those who read various versions of this thesis and provided invaluable comments: Professor Samuel Kaski, Professor Jaakko Hollmén, Lic. Tech. Esa Alhoniemi and Dr. Ignazio Diaz, and the reviewers Dr. Jari Kangas and Professor Jouko Lampinen. Also invaluable has been the support and the spirit of comradeship of the Intelligent Data Engineering research group: Jussi Ahola, Johan Himberg, Sampsa Laine, Jukka Parviainen, and other past and present members. All in all, the whole laboratory has been a place of enjoyment and inspiration. Big thanks to the whole personnel for making it so.

During the course of this work, I have participated in several projects, including the ’Adaptive and Intelligent Systems Applications’ technology program of Technology Development Center of Finland, especially the ENTIRE and ENTIRETY corporate projects with Jaakko Pöyry Consulting, the EU financed Brite/Euram project ’Application of Neural Network Based Models for Optimization of the Rolling Process’ (NEUROLL) and the ’Käyttäjäystävällinen data-analyysi’ (KÄD). I would like to express my appreciation to the financiers of and my coworkers in those projects, especially Dr. Petri Vasara and Mr. Markus Siponen from Jaakko Pöyry Consulting.

The financial support of the Finnish Foundation for the Promotion of Technology, the Finnish Foundation for Commercial and Technical Sciences, Kauhajoen kulttuurisäätiö, and the Finnish Cultural Foundation is gratefully acknowledged.

Most of all, I would like to dedicate this thesis to my dear wife Riikka. Special thanks to you for your support and forbearance.

List of publications

This thesis consists of an introduction and the following publications:

Publication 1. Juha Vesanto (1997). Using the SOM and Local Models in Time-Series Prediction. In *Proceedings of Workshop on Self-Organizing Maps (WSOM'97)*, Espoo, Finland, pp. 209–214.

Publication 2. Esa Alhoniemi, Jaakko Hollmén, Olli Simula and Juha Vesanto (1999). Process Monitoring and Modeling Using the Self-Organizing Map. In *Integrated Computer Aided Engineering* Volume 6, Number 1, IOS Press, pp. 3–14.

Publication 3. Juha Vesanto (1999). SOM-Based Data Visualization Methods. In *Intelligent Data Analysis*, Volume 3, Number 2, Elsevier Science, pp. 111–126.

Unfortunately some references were missing from the original publication (from Tables 1 and 2). An Errata page can be found after the publication.

Publication 4. Esa Alhoniemi, Johan Himberg and Juha Vesanto (1999). Probabilistic Measures for Responses of Self-Organizing Map Units. In *Proceeding of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA'99)*, ICSC Academic Press, pp. 286–290.

Publication 5. Juha Vesanto and Jussi Ahola (1999). Hunting for Correlations in Data Using the Self-Organizing Map. In *Proceeding of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA'99)*, ICSC Academic Press, pp. 279–285.

Publication 6. 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, pp. 35–40.

Publication 7. Juha Vesanto and Esa Alhoniemi (2000). Clustering of the Self-Organizing Map. In *IEEE Transactions on Neural Networks*, Volume 11, Number 3, pp. 586–600.

Publication 8. Juha Vesanto (2001). Importance of Individual Variables in the k-Means Algorithm. In *Proceedings of the Pacific-Asia Conference Advances in Knowledge Discovery and Data Mining (PAKDD2001)*, Springer-Verlag, pp. 513–518.

Publication 9. Markus Siponen, Juha Vesanto, Olli Simula and Petri Vasara (2001). An Approach to Automated Interpretation of SOM. In *Proceedings of Workshop on Self-Organizing Map 2001 (WSOM2001)*, Springer, pp. 89–94.

Publication 10. Juha Vesanto and Jaakko Hollmén (2002). An Automated Report Generation Tool for the Data Understanding Phase. In *Hybrid Information Systems*, edited by A. Abraham and M. Köppen, Physica Verlag, Heidelberg, pp. 611–626.

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

Abbreviations

BMU	Best Matching Unit
CCA	Curvilinear Component Analysis
CRISP-DM	CRoss Industry Standard Process model for Data Mining
CTM	Constrained Topological Mapping
EDA	Exploratory Data Analysis
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GTM	Generative Topographic Mapping
HSV	Hue-Saturation-Value color model
ICA	Independent Component Analysis
KDD	Knowledge Discovery in Databases
LVQ	Learning Vector Quantization
MDS	Multi-Dimensional Scaling
PCA	Principal Component Analysis
pdf	probability density function
RBF	Radial Basis Function network
SOM	Self-Organizing Map
STVQ	Soft Topographic Vector Quantization
TS-SOM	Tree-Structured Self-Organizing Map
VQ-P	Vector Quantization and Projection

Notations

$\alpha(t)$	learning rate at time t
$[\alpha_i, \beta_i]$	range of values allowed for variable i in a rule
β	a noise or variance parameter
b_i	index of the best-matching unit for a data vector $\mathbf{x}_i$
C	the number of clusters
C_i	cluster i, or the collection of data vectors belonging to cluster i: $\{\mathbf{x} \mid \mathbf{x} \in C_i\}$
$\delta(t)$	neighborhood radius at time t
$\mathcal{D}_i$	set of distances $\|\mathbf{m}_i - \mathbf{m}_j\|$ of unit i to neighboring map units $j \in \mathcal{N}_i$
d	input space dimension
d_{ij}	distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in the input space
d'_{ij}	distance between the projections of vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in the output space
$d(C_i, C_j)$	distance between clusters C_i and C_j
E	some error or cost function
h_{ij}	neighborhood kernel centered on unit i and evaluated at unit j
H_i	the sum of neighborhood function values for unit i: $H_i = \sum_j h_{ij}$
k	number of quantization points in a k-means algorithm
k_j	number of effective quantization points for variable j
M	the number of map units / prototypes
$\mathbf{m}_i$	prototype vector of unit i
m_{jk}	kth component of the prototype vector of unit j
μ_j	mean of variable j
N	the number of data samples
N_i	number of data samples in Voronoi region V_i or in cluster C_i
$\mathcal{N}_i$	the set of neighboring map units of unit i
$\mathbf{n}_i$	centroid of Voronoi set or cluster i
$p(\mathbf{x})$	probability density function of random variable $\mathbf{x}$
P^c	confidence in characterizing rule
P^d	confidence in differentiating rule
q_j	quantization quality of variable j
R	a rule
R^c	characterizing rule
R^d	differentiating rule

$\mathbf{r}_i$	location of neuron i in the output space (on the map grid)
$r(\mathbf{x}_i, \mathbf{m}_j)$	response of map unit j to data vector i
σ_j	standard deviation of variable j
$S(C_i)$	within-cluster distance measure in cluster C_i
s_{ij}	significance of variable i in cluster j
V_i	the Voronoi region around unit i
$\mathbf{x}$	a vector in the input space
$\mathbf{x}_i$	a sample vector i from the input data set
x_{ik}	kth component of sample vector i
x_{ik}^z	Z-score of kth component of sample vector i

Contents

Chapter 1

Introduction

1.1 Background

Data mining refers to the application of a wide array of methods — ranging from relational learning to statistics and neural networks — to process and analyze data [5, 32, 45]. The purpose is to find knowledge from databases where the dimensionality, complexity, or amount of data is prohibitively large for manual analysis. This is an interactive process which requires that the intuition and background knowledge of application experts are coupled with the computational efficiency of modern computer technology.

Data mining has its roots in various scientific disciplines, but the most notable root is the field of exploratory data analysis (EDA) [51, 130] in statistics in the 70's. Back then, the term "data mining" had a negative nuance. It referred to the danger of finding patterns from data even if none existed: if one keeps looking long enough, something is bound to come up.

In the late 80's and especially in the 90's the term was given a different meaning. Advances in computing and digital storage technology made it feasible to create huge databases, and the term was an appropriate slogan for the task of finding the "golden nuggets" from them. As an independent research area data mining arose in the 90's, and as a recognized industry it is only now beginning to be established [109]. Strictly speaking, data mining is just a part of Knowledge Discovery in Databases (KDD) [31], but often (and in this thesis as well) the two terms are used synonymously.

The Self-Organizing Map (SOM) [80] is a neural network algorithm that is based on unsupervised learning. It was first proposed by Academician Teuvo Kohonen in 1981 as a visualization tool [77], and has since become one of the most popular neural network methods.

The data domain of the basic SOM is numerical table-format data. Such data can be presented as a matrix which has N rows each of which is one d-dimensional data sample $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]$. Each sample corresponds to one object, for example a time-point, person or a paper machine, and the vector components are (numerical) values of a fixed set of features of that object[1]. This kind of data is frequently encountered in process analysis, social sciences, and economics, to name a few application areas.

[1]A recently introduced extension of the SOM based on generalized median is applicable to a much larger data domain: it is sufficient if there exists a distance measure between objects [85].

The SOM quantizes the training data into a representative set of prototype vectors and maps them on a low-dimensional grid. The point density of the prototypes follows roughly the probability density of the data. The training algorithm is simple, robust to missing values, and — perhaps most importantly — it is easy to visualize the map. These properties make SOM a prominent tool in data mining, especially in its initial exploratory phase.

The use of the SOM in exploratory data analysis has been previously studied in [64], where an overview of the properties of the SOM from the data exploration point of view is given, and it is compared to various related algorithms, especially projection algorithms. This thesis concentrates more on clustering using the SOM, and gives a more detailed description of how it can be used to characterize properties of the data.

The motivation for this work has come from a number of practical data mining projects, where SOM has been a central data analysis tool. For the author, the original inspiration came from a project where world's forest industry was investigated. The investigated database contained technical information of over 4000 pulp and paper mills of the world, and over 8000 pulp lines and paper machines in them. A SOM-based data analysis tool was implemented, and a clustering of pulp and paper mills, and forest industry companies was made [137, 143].

In that project, it became apparent that while the SOM could be used to quickly create a qualitative overview of the data, turning this qualitative information to quantitative characterizations required a great deal of expertise and manual work. A conclusion was that a set of post-processing methods to simplify and summarize the resulting SOM would be needed [124], see Figure 1.1.

A driving goal of this work has been to construct a framework where an overview and initial analysis of the data can be executed automatically, without user intervention. Although such analysis cannot take into account any special features of a specific data mining project, it can considerably reduce the workload of the data miner by automating many often-repeated procedures. In such context, the framework itself is of more importance than its individual parts. More important than optimality in a certain situation is that the applied analysis methods are robust, and that the produced results are "good enough" in a wide array of tasks and data sets.

1.2 Contributions

1.2.1 Data exploration framework

The main task for SOM in data mining is to act as an exploration tool for acquiring an understanding, and for generating hypothesis about, the properties of the data. Compared to the many alternative algorithms, the strength of SOM is its versatility. It can be easily used to provide information about the basic characteristics of the data:

- Statistical properties of individual variables.
- Dependencies and relationships between variables.
- Clusters or natural groups in the data, and the properties of those clusters.

In order to make this information explicit, however, the SOM needs to be visualized or post-processed in some other way. The primary aim of this work has been to investi-

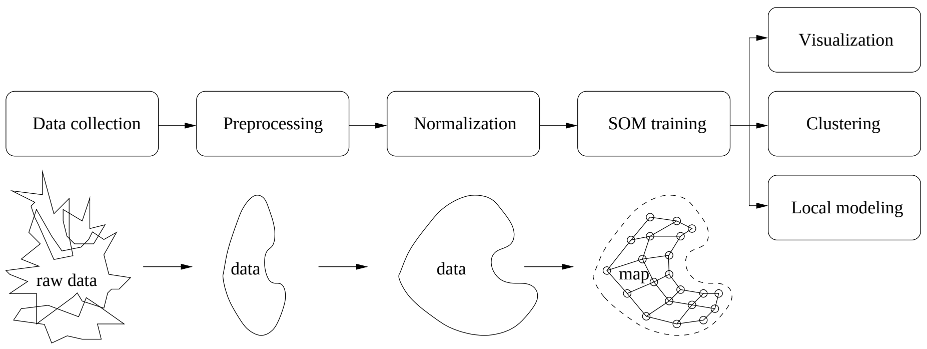


Figure 1.1: Applying the SOM in data mining. After data collection, the data is preprocessed, normalized, and a SOM is trained. This thesis concentrates on the processing that happens after training the SOM: visualization, clustering and constructing local models.

gate these post-processing methods, propose enhancements to them, and thus provide a better understanding of how the SOM can and should be used for data exploration.

A major contribution of this thesis is to construct a SOM-based data exploration framework. The core application areas of the SOM in data exploration are identified and presented as components of this common framework. It is shown how the framework relates to the overall data mining process, and the core components are compared to other, closely related methods to gain a better understanding of the possibilities, strengths, and weaknesses of the SOM in data exploration.

In the publications, the validity of SOM-based data exploration methods are investigated, and various enhancements are proposed. Instead of a deep analysis of a single application area, an attempt has been made to cover all core areas of the SOM-based exploration process. A software library for SOM training, visualization and analysis has been implemented. In addition to the results in the publications, a dis-composition of the SOM distortion measure into component parts is presented in this introduction. Below, the contents and contributions of the publications are discussed in more detail.

1.2.2 Publications

In Publication 1 (Vesanto 1997), local modeling of a chaotic time-series is investigated. The data is first quantized using SOM, and then simple linear models are constructed locally for each map unit using the data in the map unit and in its neighbors. The novelty with respect to earlier combinations of SOM and local linear models is how the local data sets are constructed. Another difference is that in this work, the models are constructed afterward, rather than simultaneously with training the SOM. This approach reduces the computational complexity of training the SOM as well as frees one from having to choose the predicted variable beforehand.

In Publication 2 (Alhoniemi, Hollmén, Simula and Vesanto 1999), the basic methodology of applying SOM for data analysis is presented. Possible applications in process monitoring and modeling are given, and three illustrative case studies are described. The author was responsible for the pulp and paper mills case study and describing validation, interpretation and visualization methods for the SOM.

In Publication 3 (Vesanto 1999), an overview and categorization of SOM-based data visualization methods is presented. The categorization indicates the range of visualization tasks that the SOM is suitable for, and links the SOM to other visualization methods. In addition, some novel enhancements to existing methods are proposed: component plane reorganization (further investigated in Publication 5) and several ways to visualize the position of a data sample on the SOM.

Publication 4 (Alhoniemi, Himberg and Vesanto 1999) deals with forming a kernel density estimate enabling one to quantify the response of the SOM units to presented data samples in a probabilistic manner. A Gaussian kernel is inserted at each map unit, and its covariance matrix is estimated using the local data. The main novelty with respect to earlier methods is in how the local data sets are augmented from the neighbors of the map unit. Three different ways to do this are tested, and the density estimates are compared with those produced by Gaussian Mixture Models (GMM) and by S-Map algorithm. The author was responsible for the original idea, took part in designing and executing the tests and writing the publication.

In Publication 5 (Vesanto and Ahola 1999), detection of correlations between variables using SOM is investigated. A simple method — component plane reorganization — is presented to enhance this task in the case of a large number of variables. Essentially, a representation for the variables is selected, for example the correlation coefficients between variables, and these are projected on a plane. Different variations of the method are evaluated on a complex test data, and an application to the analysis of data from a hot strip steel mill is presented. The author was responsible for the idea, design of the experiments, and writing the paper.

In Publication 6 (Vesanto, Himberg, Alhoniemi and Parhankangas 1999) a software package is introduced which implements SOM training, visualization and analysis algorithms in Matlab computing environment[2]. Its performance in terms of computational load and memory consumption is evaluated and compared to a corresponding C-program. The author was responsible for coordination of the software development, implementation of elementary training and analysis routines, the performance evaluation and was mainly responsible for writing the paper.

In Publication 7 (Vesanto and Alhoniemi 2000), an overview of SOM-based clustering methods is presented. The role of clustering in data exploration is discussed, and a novel method for pruning hierarchical clustering trees is proposed. Using SOM for clustering is essentially a two-phase approach: at the first phase, the SOM pre-clusters the data, and in the second phase the SOM units are clustered. Two different approaches for clustering the SOM are presented and compared with clustering the data directly. The most important benefit of the two-phase approach is a considerable decrease in computational load, making clustering of large data sets feasible. The paper was joint work, but the author had main responsibility of agglomerative clustering of the SOM and the comparison between direct and two-level clustering approaches, whereas the second author was responsible for partitional clustering.

In Publication 8 (Vesanto 2001), the problem of scaling variables in vector quantization is investigated. The error of *k*-means algorithm is investigated in terms of variables, and is shown to depend on three factors: variance, distribution characteristics, and dependencies with other variables. Two novel measures of representation quality of individual variables are proposed. Both measures are invariant with respect to variance.

In Publication 9 (Siponen, Vesanto, Simula and Vasara 2001), methods for interpretation of SOM clusters and a framework for (semi)automated analysis of hierarchical

[2]By MathWorks, Inc. `http://www.mathworks.com`

data is presented. In this framework, the clusters are derived automatically, and then characterized by ranking the variables, and by constructing characterizing rules for the variables. For forming the characterizing rules, a novel measure of significance is proposed. In case of hierarchical data, the clusters form new variables for the upper level data, and the characterizations allow one to give them meaningful names. The author was responsible for the rule generation algorithm and for writing most of the paper.

In Publication 10 (Vesanto and Hollmén 2002), the outline of a system for automatically generating data survey reports of table-format numerical data is described. The focus of the paper is on constructing a cluster hierarchy and on describing the clusters. Novel algorithms for doing these tasks are proposed. In the system, different methods and representations are combined to produce a unified and comprehensive report of the properties of the data manifold. A case study illustrating the usefulness of the report using real-world data is also presented. The author had main responsibility of design and implementation of the reporting system as well as writing the paper.

1.3 Organization

In this chapter, the background, goals and publications associated with this work have been presented. The rest of the thesis is organized as follows. In Chapter 2, the data mining process in general is shortly presented, and the role of data exploration in it is discussed. In Chapter 3, the SOM algorithm and its relationships to other algorithms are discussed. In Chapter 4, some data exploration methods for visualization and clustering of data are presented. The use of SOM for these tasks is discussed and compared to other methods. In addition, a short introduction to SOM-based local modeling is given. In Chapter 5, process models for automating data exploration of table-format data are presented, and an autonomous report generation system is described. The work is concluded in Chapter 6.

Throughout the thesis, two table-format data sets called “system” and “mills” are used to illustrate the presented methods. The system data is a relatively simple 9-dimensional data set measuring the disk, CPU and network performance of a workstation in a network environment. The mills data is the pulp and paper mill data set that was the original inspiration for this work. The data sets are described in more detail in Appendix A.

Chapter 2

Data mining

Data mining is essentially a problem-driven process: there is a question that needs an answer, or a problem that needs a solution. The answer is sought by analyzing available data. Data analysis forms the core of data mining, but the whole data mining process covers also related issues such as defining the actual business problem and deploying the solution to solve it.

This thesis concentrates on the initial exploratory phase of data mining. The aim is to provide a description of the most important properties of a given data set. The SOM-based methods offer one possible path from the data to its characterization.

The other aim is to provide the description automatically, such that the workload of the data miner is reduced. To do this, the usually ad hoc type of data exploration needs to be formalized. This chapter provides the background and basic framework for a formal data exploration process. Chapter 5 describes the process in more detail and integrates the methods discussed in Chapters 3 and 4 with it.

2.1 Data mining process

To facilitate the data mining process, there have been attempts to formalize it by breaking it into a number of sequential phases [12, 16, 31, 111]. Although the names and contents of these phases differ slightly, the same overall ideas are present in all process models: first the data miner familiarizes him/herself with the problem and the data, then the data is prepared and models are built and evaluated. Finally, the new knowledge is consolidated and deployed to solve the problem.

One of the most advanced data mining process models is the CRoss Industry Standard Process model for Data Mining (CRISP-DM) [16], illustrated in Figure 2.1. It has been developed and endorsed by a consortium of some of the major companies in data mining industry[1] and it covers the whole life-cycle of a data mining project. In CRISP-DM, the data mining process is divided to six phases: business understanding, data understanding, data preparation, modeling, evaluation and deployment:

- *Business understanding:* At first, the data miner familiarizes him/herself with the problem domain. Domain knowledge, or business understanding, is important in all phases of data mining. It is impossible to make decisions about what to ignore and what to pursue further without appropriate knowledge of what is interesting,

[1]Most notably SPSS, Inc. `http://www.spss.com`

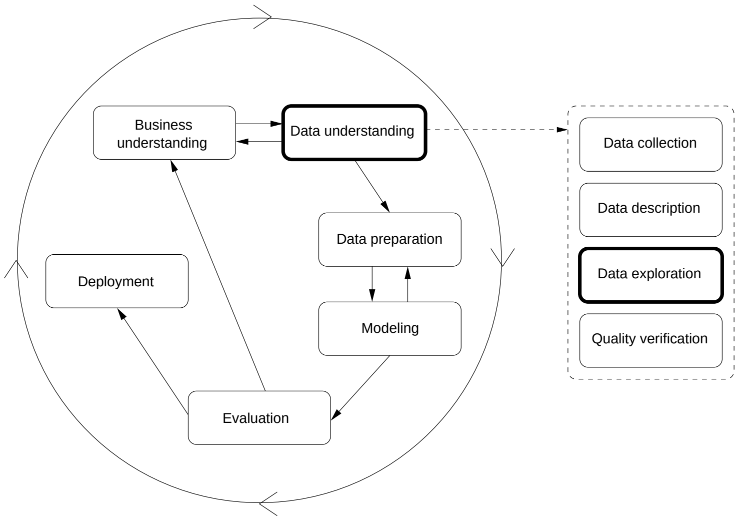


Figure 2.1: CRISP-DM: process model for data mining [16]. The data understanding phase is further divided to four sub-tasks. The boxes outlined with thicker lines indicate the focus of this thesis.

surprising, or relevant with respect to the problem that is being solved. Without the necessary knowledge, the miner is just trashing around in the dark.

- *Data understanding:* Data understanding includes both understanding the origin, nature and reliability of the data, as well as becoming familiar with the contents of the data through data exploration. Proper data preparation, selection of modeling tools and evaluation processes is only possible if the miner has a good overall idea, a mental model, of the data.

  The CRISP-DM divides the data understanding phase to four sub-tasks:

  – *Collection of data* starts with finding out what kind of data is available and how it can be accessed. As opposed to experimental set-ups, in data mining the data usually exists even before the data mining process starts. However, it has probably been gathered for purposes other than data analysis, and it may have been stored in multiple locations in widely different formats. Gathering and combining the data is often very time-consuming.

  – Construction of a *data description* provides the data miner with a documentation of both meta-knowledge — such as origin, physical source, and access methods and protocols for the data — and rough "surface" properties of the data, for example simple descriptive statistics.

  – *Data exploration* continues from where data description left off. Its purpose is to evaluate possibilities present in the data as well as to acquire an overall idea, a mind model or a map, of the data manifold: what are the typical values, what values can be considered unusual and how different values (or variables) are related to each other?

  – *Quality verification* assesses whether the data is reliable and complete enough with respect to the problem. The aim is to identify possible problems in the data: missing values, noise, inconsistencies, and sampling bias.

- *Data preparation:* The fundamental aim of data preparation is to make it easier to build precise and reliable models by correcting errors and extracting new features. Data preparation is a diverse and difficult issue. It is so application dependent that only some general guidelines for it can be given [111].

- *Modeling:* Modeling is the phase where the solution to the problem is sought. The previous phases are basically preparation for modeling and the later phases deal with its deployment in practice, but the actual solution is specified at this phase. The solution may be a predictive model, or more descriptive in nature: a segmentation or clustering of the data into a set of distinct groups, analysis of certain properties like dependencies between variables, or an estimate of the probability density function of the data. Modeling is perhaps the most thoroughly discussed issue in the literature, see for example [8, 18, 20, 114].

- *Evaluation:* Before deploying the solution, it needs to be evaluated from the point of view of the original business problem: to determine whether the found solution is good enough to be deployed. Note that besides the solution, the data mining process generates insights, ideas, and secondary models. These are also important with respect to the business problem.

- *Deployment:* Finally, the solution is employed to solve the original problem.

In practice, the data mining process is highly iterative. Any phase may raise questions or ideas that need to be investigated or implemented in an earlier phase. Consider, for example, the loops between business and data understanding, and data preparation and modeling in Figure 2.1. Both loops may require considerable iteration to determine what kinds of patterns are interesting or relevant with respect to the business problem, and how they can be extracted from the data.

2.2 Data understanding and exploration

This thesis is mainly concerned with the data understanding phase of CRISP-DM. This phase can be found in most other data mining process models, too, albeit under a different name. In [111], Pyle introduces the concept of data survey for getting a feel of the data manifold. In Brachman's process model [12], a phase called "data discovery" serves the same purpose. The knowledge discovery process in [31] does not explicitly list a data understanding phase. However, the corresponding task is closely integrated into the loop of data preparation and analysis.

The goal of data understanding is to get a feel for what the data looks like, what situations it covers (and what not), and how reliable it is. The methods do not aim at making a precise model of the data but rather making sense of it. After gaining holistic understanding of the data as a whole, it is possible to return and inspect the details more carefully.

The core of data understanding, and the focus of this thesis, is exploratory data analysis. A high-level view of the data exploration process is presented in Figure 2.2. The process consists of preprocessing, analysis and review phases.

- In the preprocessing phase, the original raw data is cleaned and transformed so that it presents interesting data properties more clearly, has no or at least fewer erroneous values, and is in a form suitable for the subsequent analysis methods.

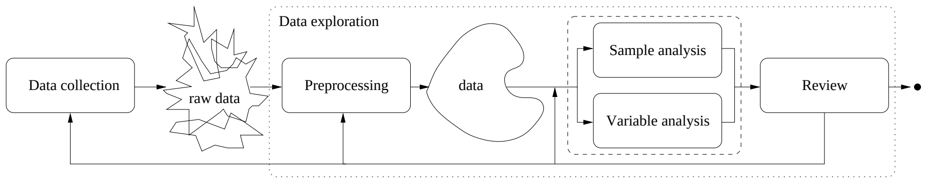


Figure 2.2: Data exploration process for table-format data (inside the box drawn with a dotted line). The preprocessing–analysis–review loop is close to the preparation–modeling loop in CRISP-DM. In fact, the difference between data exploration and the actual data analysis is often vague. Rather, they can be considered two ends of the same line.

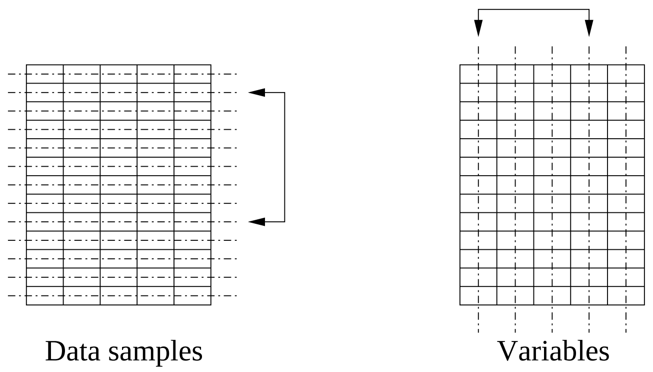


Figure 2.3: Table-format data can be investigated both in terms of samples and in terms of variables. Typically, the number of samples is much higher than the number of variables.

- After preprocessing, analysis methods are applied. Numerical table-format data can be explored from two viewpoints, see Figure 2.3. On one hand, the table can be considered horizontally, as a collection of data samples, where the similarities between individual data objects are considered interesting. On the other hand, the table can be investigated vertically, as a set of variables the statistical properties and dependencies of which are the point of interest.
- Finally, the analysis results are reviewed, and the data miner decides whether to return to some earlier phase, or to exit from the exploration loop.

A basic tenet of exploratory data analysis is to simply take a look at the data to judge its nature and complexity. This is done by interactive browsing, for example using visualizations, and through informative descriptive measures of the properties of the data. To produce the visualizations and summaries, measures and algorithms from statistics, artificial neural networks, rule induction, and various other disciplines are used. In the case of table-format data, especially important are projection and clustering methods for the analysis of data samples, and multivariate statistics for the analysis of dependencies between variables. Other data domains, for example categorical data, text analysis and time-series analysis have their own exploratory algorithms. However, these algorithms are out of the scope of this thesis.

Due to the iterative nature of the data mining process, several different data sets and preprocessing strategies need to be considered and explored. One goal of this thesis is to make the process less time-consuming by automating parts of it. In Publication 10, a system is described which generates data survey reports autonomously allowing the user to move directly from the data preprocessing to the review step. Because of the highly iterative nature of the data exploration process — and the whole data mining process in general — such a system can reduce the active working time needed from the data miner considerably.

Chapter 3

Self-Organizing Map

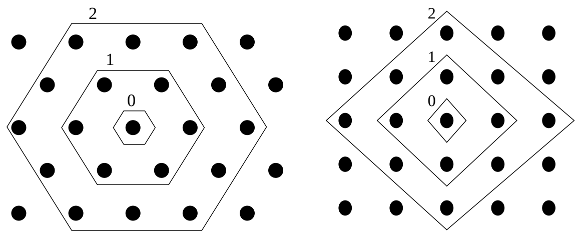
The Self-Organizing Map (SOM) is one of the most widely used neural network methods [80]. It is mainly used for visualization and clustering of data. The applications range from process monitoring [122] to organization of document collections [82]. In [22] a number of data analysis cases related to economics are presented in which the SOM has been an important tool. More examples of fruitful usage of the SOM in various engineering tasks can be found for example in [84, 123]. A comprehensive bibliography of SOM research until 1997 has been compiled by Kaski *et al.* [66].

The basic SOM algorithm described in this chapter is fairly well established. The main contributions of this thesis deal with post-processing a trained SOM (Chapter 4), and with formulating the SOM-based data exploration process (Chapter 5). In addition, this chapter presents two original contributions. In Section 3.1.2, the SOM distortion measure is divided into three component parts which offer some insight into the nature of the SOM algorithm. This result has not been published elsewhere. In Section 3.1.3, an important but often ignored viewpoint into the quantization problem is discussed: how properties of individual variables effect the quantization result. This has been investigated in Publication 8.

3.1 The basic SOM

The basic SOM consists of M units located on a regular low-dimensional grid, usually 1- or 2-dimensional, see Figure 3.1. Higher dimensional grids are possible, but they are not generally used since their visualization is problematic[1]. Each unit j has an associated d-dimensional prototype vector $\mathbf{m}_j = [m_{j1}, \ldots, m_{jd}]$. The unit positions $\mathbf{r}_j$ on the grid are fixed from the beginning. The map adjusts to the data by adapting the prototype vectors. Together the grid and the set of prototype vectors form a low-dimensional map of the data manifold: a 2-dimensional representation where topologically closely related objects (map units) are close to each other.

[1]There are, of course, exceptions. For example, Kiviluoto has visualized 3-dimensional map grids successfully [72]. If visualization is not needed, even higher than 3-dimensional grids may be beneficial [126].



(a) Hexagonal grid (b) Rectangular grid

Figure 3.1: Two SOMs of size 5 × 5 units: (a) hexagonal lattice, (b) rectangular lattice. The lines indicate neighborhood sets $\mathcal{N}_i = \{j \mid \|\mathbf{r}_i - \mathbf{r}_j\| \leq \sigma\}$ at different neighborhood radius values ($\sigma = 0$, 1, and 2). In this thesis, unless otherwise specified, $\sigma = 1$ is used to define the neighborhood of each unit.

3.1.1 Training algorithm

Sequential training algorithm

The SOM algorithm is iterative. At each training step t, a sample data vector $\mathbf{x}_i$ is randomly chosen from the training set. Distances between $\mathbf{x}_i$ and all the prototype vectors are computed. The best-matching unit (BMU), denoted here by b_i, is the map unit with prototype closest to $\mathbf{x}_i$:

$$b_i = \arg\min_j \{\|\mathbf{x}_i - \mathbf{m}_j(t)\|\}. \tag{3.1}$$

If the sample vector $\mathbf{x}_i$ has some missing values, those variables are ignored in the distance calculation and in the subsequent update step.

Next, the prototype vectors are updated by moving them toward $\mathbf{x}_i$, as shown in Figure 3.2. The update rule for the prototype vector j is:

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \alpha(t) h_{b_i j}(t)[\mathbf{x}_i - \mathbf{m}_j(t)], \tag{3.2}$$

where t is the training step index, $\alpha(t)$ is learning rate and $h_{b_i j}(t)$ is a neighborhood kernel centered on the winner unit. The kernel gets its biggest value for the winner unit, and decreases monotonically with increasing distance on the map grid $\|\mathbf{r}_{b_i} - \mathbf{r}_j\|$. The kernel can be for example Gaussian:

$$h_{b_i j}(t) = e^{-\frac{\|\mathbf{r}_{b_i} - \mathbf{r}_j\|^2}{2\sigma^2(t)}}, \tag{3.3}$$

where $\mathbf{r}_{b_i}$ and $\mathbf{r}_j$ are positions of units b_i and j on the SOM grid and $\sigma(t)$ is neighborhood radius. Both the learning rate $\alpha(t)$ and the neighborhood radius $\sigma(t)$ decrease monotonically during training, learning rate to zero, and neighborhood radius to some suitable nonzero value, typically one. In the rest of the thesis, the step index t will be left out for sake of brevity.

During training, the SOM behaves like a flexible net that folds onto the "cloud" formed by the training data. Because of the neighborhood relations, neighboring prototypes are pulled to the same direction, and thus neighboring units acquire similar prototype vectors.

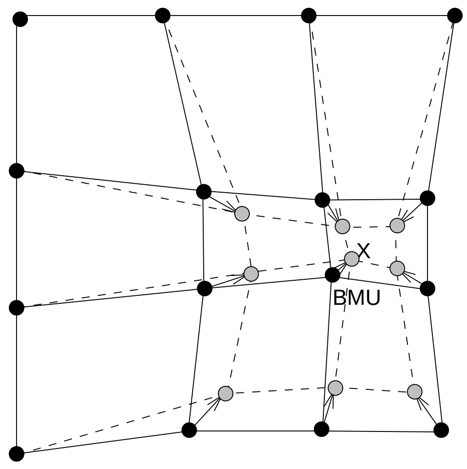


Figure 3.2: Updating the best matching unit (BMU) and its neighbors toward the input sample marked with x. The black and gray circles correspond to situation before and after updating. The solid and dashed lines show neighborhood relations, respectively.

The prototype vectors define a tessellation of the input space into a set of Voronoi regions or Voronoi sets $V_j = \{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{m}_j\| < \|\mathbf{x}_i - \mathbf{m}_k\| \forall k \neq j\}$, each corresponding to one map unit. In effect each data vector belongs to the Voronoi set of the map unit to which it is closest.

Batch map

Batch map [80] is a version of the SOM based on fixed point iteration. In each training step, the BMUs b_i of all data vectors are found at first as in Eq. 3.1. After this, the new prototypes are calculated as:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} h_{b_i j} \mathbf{x}_i}{\sum_{i=1}^{N} h_{b_i j}}. \tag{3.4}$$

The new prototypes are weighted averages of the data samples, such that the weight of each data sample is the neighborhood kernel value $h_{b_i j}$ at its BMU b_i. The basic SOM is a stochastic approximation of the Batch map algorithm.

Alternatively, the update can be calculated as a weighted average of the Voronoi set centroids $\mathbf{n}_j = \frac{1}{N_j} \sum_{\mathbf{x}_i \in V_j} \mathbf{x}_i$:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{M} N_i h_{ij} \mathbf{n}_i}{\sum_{i=1}^{M} N_i h_{ij}}, \tag{3.5}$$

where N_j is the number of samples in Voronoi set V_j. This allows a much more efficient matrix-based implementation than using Eq. 3.4 directly [138, 141].

3.1.2 Quality measures

Distortion measure

In order to understand the SOM algorithm, it is important to know what kind of SOMs are "good". For many algorithms, a cost or energy function exists which explicitly defines the optimal situation. However, for the basic SOM algorithm it has been shown that it is not the gradient of any cost function in the general case [29]. In case of a discrete data set and fixed neighborhood kernel, the map distortion measure

$$E_d = \sum_{i=1}^{N} \sum_{j=1}^{M} h_{b_i j} \|\mathbf{x}_i - \mathbf{m}_j\|^2 \tag{3.6}$$

can be shown to be a local cost function of the SOM [78]. When the BMU index b_i of any of the data samples $\mathbf{x}_i$ changes, the cost function changes slightly, and thus the SOM only gives an approximate solution to Eq. 3.6. If one is only interested in minimizing Eq. 3.6, the solution can be obtained by changing the definition of winner (Eq. 3.1) to

$$b_i = \arg\min_j \{\sum_k h_{jk} \|\mathbf{x}_i - \mathbf{m}_k\|^2\}, \tag{3.7}$$

but this is computationally much heavier than the basic SOM [48].

In [93], Lampinen and Kostiainen propose a generative probability density model which is consistent with the distortion measure. The probability density is a mixture of cut Gaussians each existing within one Voronoi cell. The density is discontinuous on the borders of the Voronoi cells. Inside a Voronoi region, the density function has the form:

$$p(\mathbf{x} \,|\, \mathbf{x} \in V_j) = Z e^{-\beta W_j - \frac{\|\mathbf{x} - \bar{\mathbf{m}}_j\|^2}{2 s_j^2}} \tag{3.8}$$

where Z is a normalization constant, and β is a variance parameter which is set after training the SOM such that the likelihood of the data is maximized. The other parameters are defined as:

$$\begin{aligned} \bar{\mathbf{m}}_j &= \frac{\sum_k h_{jk}\mathbf{m}_k}{\sum_k h_{jk}} = \frac{\sum_k h_{jk}\mathbf{m}_k}{H_j} && (3.9) \\ s_j^2 &= \frac{1}{2\beta H_j} && (3.10) \\ W_j &= \sum_k h_{jk} \|\mathbf{m}_k - \bar{\mathbf{m}}_j\|^2 && (3.11) \end{aligned}$$

where $H_j = \sum_k h_{jk}$ (see Figure 3.3). Notice that $\bar{\mathbf{m}}_j$ is the weighted mean of the prototype vectors, where the neighborhood function values h_{jk} are used as weighting factors: $\bar{\mathbf{m}}_j = E_h\{\mathbf{m}|j\}$. Equally, W_j can be interpreted as a weighted total variance of the prototype vectors: $W_j = H_j \mathrm{Var}_h\{\mathbf{m}|j\}$. Using these terms, Eq. 3.8 can be written as:

$$p(\mathbf{x} \,|\, \mathbf{x} \in V_j) = Z e^{-\beta H_j (\mathrm{Var}_h\{\mathbf{m}|j\} + \|\mathbf{x} - \bar{\mathbf{m}}_j\|^2)}. \tag{3.12}$$

Elements of the distortion measure

The distortion measure E_d can be divided to two component parts [95, 64]:

$$E_d = \sum_{i=1}^{N} H_{b_i} \|\mathbf{x}_i - \mathbf{n}_{b_i}\|^2 + \sum_{i=1}^{M} N_i \sum_{j=1}^{M} h_{ij} \|\mathbf{n}_i - \mathbf{m}_j\|^2, \tag{3.13}$$

where b_i is the index of the BMU of data vector $\mathbf{x}_i$, N_i is the number of data items in the Voronoi set V_i of unit i, and $\mathbf{n}_i$ is their centroid $\sum_{\mathbf{x} \in V_i} \mathbf{x}/N_i$. If the neighborhood function values for each map unit are normalized to unity such that $H_i = 1$, $\forall i$, the first

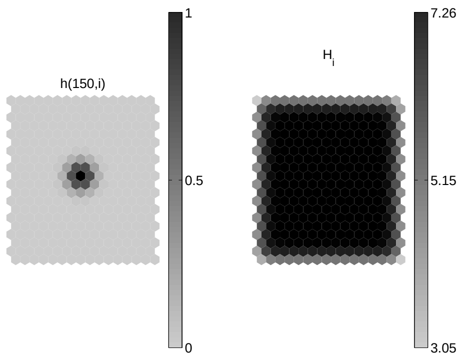


Figure 3.3: Neighborhood function values for the unit in the center (on the left), and the sum of neighborhood function values H_i for each map unit (on the right).

term corresponds to classical vector quantization error (see Section 3.1.3). It can also be expressed as a sum over the variances in each Voronoi set:

$$\sum_{i=1}^{N} H_{b_i} ||\mathbf{x}_i - \mathbf{n}_{b_i}||^2 = \sum_{j=1}^{M} H_j \sum_{\mathbf{x}_i \in V_j} ||\mathbf{x}_i - \mathbf{n}_j||^2 = \sum_{j=1}^{M} H_j N_j \mathrm{Var}\{\mathbf{x}|j\}.$$

The second term can be further divided to two parts such that the distortion measure can be expressed as a sum of three component parts $E_d = E_{qx} + E_{nb} + E_{nv}$:

$$\begin{aligned} E_d &= \sum_{j=1}^{M} N_j H_j \Big(\mathrm{Var}\{\mathbf{x}|j\} + ||\mathbf{n}_j - \bar{\mathbf{m}}_j||^2 + \mathrm{Var}_h\{\mathbf{m}|j\}\Big) \\ &= \underbrace{\sum_{j=1}^{M} N_j H_j \mathrm{Var}\{\mathbf{x}|j\}}_{E_{qx}} + \underbrace{\sum_{j=1}^{M} N_j H_j ||\mathbf{n}_j - \bar{\mathbf{m}}_j||^2}_{E_{nb}} + \underbrace{\sum_{j=1}^{M} N_j H_j \mathrm{Var}_h\{\mathbf{m}|j\}}_{E_{nv}}. \end{aligned} \tag{3.14}$$

The term E_{qx} measures the quantization quality of the map. Notice that it is the only term that involves the data vectors $\mathbf{x}_i$ directly. The last term E_{nv} corresponds to the ordering quality of the map by measuring the weighted variance of the prototype vectors in the neighborhood. It gets lowest values when the prototype vectors are as close to each other as possible. In terms of E_{nv}, the ideal placement for the prototype vectors is one where they form a regular, as tightly packed grid as possible. The middle term E_{nb} measures the neighborhood-induced bias in quantization. It can be interpreted as the stress between quantization and ordering properties.

Other quality measures

In addition to E_d, several other quality measures have been proposed and used for the SOM. The quantization property is usually measured using quantization error:

$$E_q = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{m}_{b_i}||^2. \tag{3.15}$$

Other measures, which also take the neighborhoods into account have been proposed in [21, 42, 63, 73, 136, 144, 145].

3.1.3 Vector quantization

Classical vector quantization

Vector quantization algorithms [39] try to find a set of prototype vectors $\mathbf{m}_i$, $i = 1, \dots, M$ which reproduce the original data set as well as possible. The best known algorithm to find these prototypes is the k-means algorithm [107]. It finds a set of $M = k$ prototype vectors which minimize the quantization error:

$$E_q = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{d} |x_{ij} - m_{b_i j}|^r, \tag{3.16}$$

where b_i is the index of the best matching prototype (according to Eq. 3.1), and r is the distance norm ($r = 2$ for Euclidean metric). The point density of the prototypes follows the density of the training data. Asymptotically it holds that:

$$p(\mathbf{m}) \propto p(\mathbf{x})^{\frac{d}{d+r}} \tag{3.17}$$

where d is the dimension and $p(\mathbf{x})$ and $p(\mathbf{m})$ are the probability density functions of the input data and the prototype vectors, respectively [37, 79, 149].

Quantization reduces the original data set to a small representative set of prototypes to work with. The representative set of prototypes can be utilized in computationally intensive tasks, like clustering or projection, to get approximative results with reduced computational cost, as done in Publication 7. This reduction is important especially in data exploration. In addition, since the prototypes are formed as averages of the data samples, the effect of zero-mean noise as well as outliers are reduced.

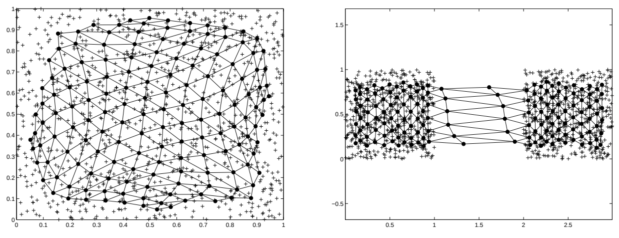
SOM as a quantization algorithm

The SOM is closely related to the k-means algorithm. If the neighborhood kernel value is one for the BMU and zero elsewhere ($h_{b_i j} = \delta(b_i, j)$ in Eq. 3.2), the SOM reduces to the adaptive k-means algorithm. Equally, the Batch Map algorithm reduces to batch k-means.

The difference between classical vector quantization and SOM is that the SOM performs local smoothing in the neighborhood of each map unit. This smoothing creates the ordering of the prototypes, but when the neighborhood radius σ is decreased during the training, it also implements a simulated annealing type of learning scheme that makes the quantization process more robust. There are also two side-effects (see Figure 3.4):

- *Border effect*. The neighborhood definition is not symmetric on the borders of the map. Therefore, the density estimation is different for the border units than for the center units of the map [80, 93]. In practice, the map is contracted on the borders. This has the effect that the tails of the marginal distributions of variables are less well presented than their centers. In some cases, this may help to reduce the effect of outliers, but in general, this is a weakness of the SOM.

- *Interpolating units*. When the data cloud is discontinuous, interpolating units are positioned between data clusters providing convenient interpolating estimates of



(a) Border effect (b) Interpolating units

Figure 3.4: Two side effects caused by the neighborhood function: (a) border effect and (b) interpolating units. The + are the training data, and the connected grid of circles is the map.

the data distribution. However, in case of some analysis tools, for example single linkage clustering, these may give false cues of the shape of the data manifold and may need to be deemphasized or completely left out of analysis, as done in Publication 7.

For SOM, a power law similar to Eq. 3.17 has been derived in one-dimensional case [116]:

$$p(\mathbf{m}) \propto p(\mathbf{x})^{\frac{2}{3} - \frac{1}{3\sigma^2 + 3(\sigma+1)^2}}, \tag{3.18}$$

where σ is the neighborhood radius on both sides of each map unit. Even though the power law holds only when the number of prototypes approaches infinity and neighborhood width is very large, numerical experiments have shown that the results are relatively accurate even for a small number of prototypes [79]. Thus, while the connection between the density of prototypes of SOM and the input data has not been derived in the general case, it can be assumed that the SOM follows at least roughly the density of the training data.

Importance of variables in quantization

One way to look at the quantization problem is in terms of variables. Some variables are more important with respect to quantization than others. By adding, removing, or rescaling variables, a different quantization result is acquired because the quantization error function E_q changes correspondingly. How well each variable is represented in the quantization depends on how strongly the variable effects the total quantization error. The importances of variables defines the "viewpoint" of the quantization. When quantization has a central role in data analysis, it is important to know what is this viewpoint, because any analysis based on the quantization will reflect how well the variables are represented.

The quantization error E_q can be expressed in terms of variable-wise errors E_j:

$$E_q = \sum_{j=1}^{d} \frac{1}{N} \sum_{i=1}^{N} |x_{ij} - m_{b_i j}|^2 = \frac{1}{N} \sum_{j=1}^{d} E_j. \tag{3.19}$$

In order to measure the granularity of the quantization with respect to each variable, the errors E_j can be compared to quantizations performed on each variable separately with increasing number of quantization points $E_j(k)$, $k = 1, 2, \ldots$. Depending on the distribution characteristics of the variable, the quantization error decreases at different rates. For example, for a uniformly distributed variable, the quantization error reduces according to formula $E_j(k) = \sigma_j^2 k^{-2}$, where σ_j is the standard deviation of the variable. Based on this, two importance measures for variables are proposed in Publication 8:

- The effective number of quantization points k_j for variable j is the minimum number of quantization points needed to achieve the variable-wise quantization error E_j when variable j is quantized independently of other variables:

  $$k_j = \arg_k \{E_j(k) = E_j\} \tag{3.20}$$

  The higher the k_j, the more emphasis the variable gets in the quantization. In principle, k_j can get only integer values, but in Publication 8 linear interpolation is used to get continuous values.

- The quantization quality q_j measures how close E_j is to minimum possible error. In a quantization with M points, the minimum quantization error for a variable is $E_j(M)$ and the maximum error is $E_j(1)$. Therefore

  $$q_j = \frac{E_j - E_j(M)}{E_j(1) - E_j(M)} \tag{3.21}$$

Notice that when a variable is easy to quantize, for example when it is binary, the value of k_j may be quite low even if q_j indicates a very high quantization quality. On the other hand, k_j gives a better expression of the granularity of the quantization with respect to each variable. Thus, in practice, both measures are useful.

Publication 8 also investigates which factors effect the importances of variables. The E_j can be expressed in terms of Z-scores. The Z-score x_{ij}^z of a variable is calculated by subtracting its mean μ_j and dividing by its standard deviation σ_j: $x_{ij}^z = \frac{x_{ij} - \mu_j}{\sigma_j}$. However, only the coefficient effects the quantization error. Therefore

$$E_j = \frac{1}{N} \sum_{i=1}^{N} |x_{ij} - m_{b_i j}|^2 = \frac{\sigma_j^2}{N} \sum_{i=1}^{N} |x_{ij}^z - m_{b_i j}^z|^2 = \sigma_j^2 E_j^z, \tag{3.22}$$

where E_j^z is the quantization error of Z-scored variable j. Thus, the importance is highly dependent on the variance or scale of the variable. However, it is not the only factor. Further insight can be gained by measuring how the quantization error E_q changes when the importance (measured by k_j) of some variable increases:

$$\frac{\Delta E_q}{\Delta k_j} = \sum_{j'=1}^{d} \frac{\Delta E_{j'}}{\Delta k_j} = \sum_{j'=1}^{d} \frac{\Delta E_{j'}}{\Delta k_{j'}} \frac{\Delta k_{j'}}{\Delta k_j} = \sum_{j'=1}^{d} \sigma_{j'}^2 \frac{\Delta E_{j'}^z}{\Delta k_{j'}} \frac{\Delta k_{j'}}{\Delta k_j}. \tag{3.23}$$

Unfortunately, these effects are hard to calculate in practice, since $E_j(k)$ can only be evaluated for integer values of k. Thus Eq. 3.23 should be regarded only as an intuitive expression which indicates that the importance of each variable in quantization depends on three factors:

1. the scale of the variable $\sigma_{j'}^2$,

2. the distribution characteristics of the variable $\frac{\Delta E_{j'}}{\Delta k_{j'}}$, and

3. the dependencies between variables: $\frac{\Delta k_{j'}}{\Delta k_j}$.

Notice that when the data is preprocessed before quantization such that all variables have the same variance, the quantization is only affected by the two latter properties. Notice also that the two proposed importance measures k_j and q_j are both invariant to the scales of the variables.

3.1.4 Vector projection

Projection methods

Projection methods try to find low-dimensional coordinates that preserve the distances (or the order of distances) between the originally high-dimensional objects. A classical projection method is multi-dimensional scaling (MDS) [89] which tries to preserve pairwise distances between all objects while reducing the dimension. The error function to be minimized is:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij} - d'_{ij})^2, \tag{3.24}$$

where d_{ij} is the distance between data samples i and j in the input space $||\mathbf{x}_i - \mathbf{x}_j||$, and d'_{ij} is the corresponding distance between the projection coordinates in the output space. There is also a non-metric version of MDS which tries to preserve the rank order of the distances [89].

In Eq. 3.24, the bigger distances have larger effect on the error function and thus are considered much more important than details in the small scale. In other projection techniques, for example Sammon's mapping [119] and Curvilinear Component Analysis (CCA) [23] the nearby samples are weighted more, and thus small distances are preserved better.

$$\text{Sammon's mapping:} \quad E = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij} - d'_{ij})^2 / d_{ij} \tag{3.25}$$

$$\text{CCA:} \quad E = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij} - d'_{ij})^2 e^{-d'_{ij}} \tag{3.26}$$

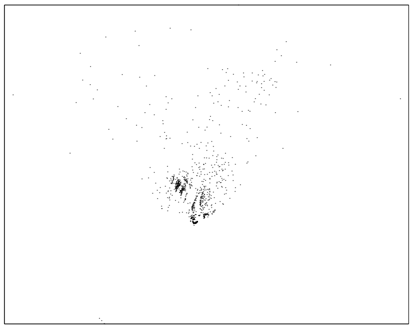
Note that Sammon's mapping emphasizes small distances in the input space, whereas CCA emphasizes output space. Examples of projections with these algorithms can be found from Figure 3.5. Visualizations based on projection methods are further discussed in Section 4.1.1.

SOM as a projection algorithm

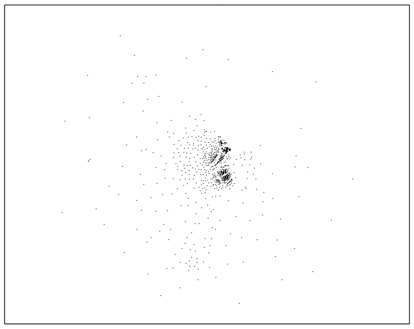
The SOM has also vector projection properties. Consider the SOM distortion measure:

$$E_d = \sum_{i=1}^{N} \sum_{j=1}^{M} h_{b_i j} ||\mathbf{x}_i - \mathbf{m}_j||^2 = \sum_{i=1}^{N} \sum_{j=1}^{M} h(d'_{ij}) d_{ij}^2. \tag{3.27}$$

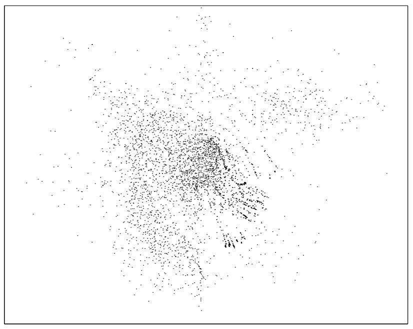
where $h(\cdot)$ is the neighborhood kernel function. Since it is usually monotonously decreasing function of d'_{ij}, small distances in the output space are emphasized, like in CCA. On the other hand, since there are more map units where there is a lot of data,



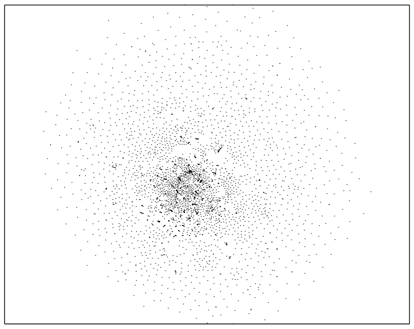
(a) System: Sammon



(b) System: CCA



(c) Mills: Sammon



(d) Mills: CCA

Figure 3.5: Sammon's mapping (a and c) and CCA projection (b and d) of the data sets. For system data, the results are qualitatively very much alike: there are two groups of tightly packed data points, both of which are further divided into several subgroups, and a cloud of more sparsely distributed data. For mills data, the correspondences between the two projections are less obvious. Note that Sammon's mapping for mills data (c) only shows the central part of the projection: only 98% of the data points are shown. The $1/d_{ij}$ nonlinearity in Eq. 3.25 makes Sammon's mapping sensitive to pairs of close-lying data points: such points are often projected very far from the rest of the points. Other projections of the data sets are shown in Figures 4.3 and 4.7.

the output space distances d'_{ij} are also dependent on the data density. Thus, the definition of locality in SOM tunes to data density as opposed to Sammon's mapping and CCA, where the definition of "small distance" is global.

Rather than try to preserve the original distances, the SOM orders prototype vectors on a predefined map grid such that local neighborhood sets in the projection are preserved. In a recent study it was noticed that the SOM is especially good at maintaining the trustworthiness of the projection: if two data samples are close to each other in the visualization, they are more likely to be close in the original high-dimensional space as well [136]. For the other aspect of neighborhood preservation — preserving the original neighborhood in the projection — the SOM was comparable to the other investigated methods, which included PCA, Sammon's mapping, non-metric MDS and GTM.

3.1.5 Computational complexity

An implementation of one epoch (going through the data once) of the SOM algorithm in C programming language is presented in Appendix B. The BMU search has $3NMd$ and the update step $3NM(d+1)$ floating point operations [138], where N is the number of data samples, M the number of map units and d is the input space dimension, so the computational complexity is $O(NMd)$. The number of training epochs multiplies this by some small number. In practice, if $N \gg 10M$, only one or two epochs of training is sufficient. Even when M is closer to N, reasonable results can be achieved in less than 10 epochs.

The memory consumption of the algorithm is $(N+M)d$ floating points for data and map prototype vector matrices (assuming that the data is in the main memory, which need not be the case). In addition, there is the matrix of inter-unit distances which is calculated beforehand. It has M^2 elements, although this can be reduced to $(M-1)(M-2)/2$ floating points since the matrix is symmetric and all elements on the diagonal are zero.

In practice, the complexity of the algorithm is mainly governed by the number of map units. If the number of map units is chosen to be proportional to $\sqrt{N}$ as in Publication 7, the complexity of the training is proportional to $O(N^{1.5}d)$. Of course, this choice is quite arbitrary. Depending on the application, the required number of map units may be independent of the number of data samples, or it may need to be directly proportional to N. In [82], the trained SOM had 10000 units, but usually maps with a few hundred map units are sufficient.

Training very large maps is time-consuming and requires a lot of memory, but the process can be speeded up with special techniques. These are basically based on speeding up the winner search by investigating only a small number of prototypes [65, 82, 86]. Such techniques can decrease the winner search from $O(Md)$ to $O(log(M)d)$. Furthermore, the training algorithm can be easily implemented in a parallel manner [81, 96]. Thus, the SOM is applicable to very large data sets.

3.1.6 Examples

Training of the SOMs of the system and mills data sets was done using the SOM Toolbox implementation of the SOM introduced in Publication 6. The Batch map algorithm was used because its implementation in Matlab is considerably more efficient than that of the basic SOM. The SOMs were initialized linearly along a plane spanned by the two biggest eigenvectors of the data. The training parameters, selected automatically,

Table 3.1: Training parameters. In the SOM Toolbox, the total number of map units is selected using a heuristic formula $M = 5\sqrt{N}$. The size of the map grid is then set according to the ratio between the two biggest eigenvalues of the covariance matrix of the data. The training itself is done in two phases: rough training (from $t = 0$ to $t = t_1$) and fine-tuning (from $t = t_1$ to $t = t_1 + t_2$).

	system	**mills**
size	[22 × 10] = 220	[20 × 16] = 320
t_1	5 epochs	5 epochs
t_2	5 epochs	5 epochs
$\sigma(0)$	3	3
$\sigma(t_1)$	1	1
$\sigma(t_1 + t_2)$	1	1
time	3.33 s	21.2 s

are listed in Table 3.1. In addition to the SOM, a quantization of the data using batch *k*-means algorithm was done. The *k*-means algorithm was initialized with the trained SOM prototype vectors, and trained for additional 10 epochs.

Table 3.2 lists relative errors at different phases of training for the SOMs, and for the *k*-means algorithm. Figure 3.6 shows the corresponding PCA-projections of the data and the prototype vectors. It can be seen that after linear initialization, the quantization error E_{qx} and neighborhood bias E_{nb} are rather high, but neighborhood variance E_{nv} is very low. During the training process, the quantization error decreases and the neighborhood variance increases such that for the system map, E_{qx} is lower than E_{nv} at the end. For the mills map, on the other hand, the quantization error remain higher than the neighborhood variance. The quantization error E_{qx} can be compared with the quantization error E_q of the *k*-means. For the system map, almost optimal quantization has been achieved.

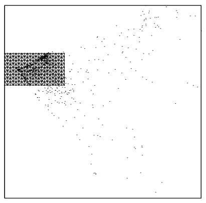
3.2 Variants and related algorithms

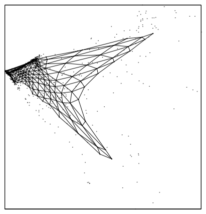
A number of variants for the SOM have been proposed (for a review, see for example [80]). The common factor to most of these variants is that they are essentially a collection of prototype vectors (or other local models) and a set of neighborhood relations defined between them. These are iteratively adjusted to correspond to the training data in such a manner that neighboring prototypes become similar to each other.

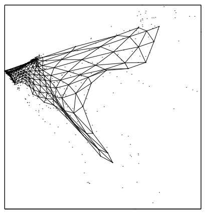
In the basic SOM, the neighborhoods are defined by giving the prototypes fixed positions $\mathbf{r}_i$ on a low-dimensional output plane. In some situations, it is a definite advantage to be able to specify the shape of the projection beforehand, but usually this is a handicap. The predefined map shape does not directly convey any useful information, and there are situations where the neighborhoods are simply wrong. In many variants — for example MST-SOM [61], neural gas [101], and GCS [34, 35] — the neighborhood relations are considerably more flexible in order to approximate the data better. However, this happens at the cost of making visualization more difficult. Also several such variants of the SOM have been proposed which exist between these two extremes: the nodes have low-dimensional positions, but this location is somewhat flexible [2, 10, 36, 58, 113, 117].

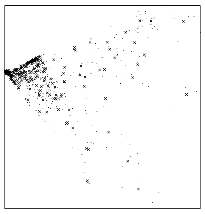
Table 3.2: Error measures for both maps at different phases of training. The error measures have been scaled by HE_1, where $H = \frac{1}{M}\sum_{i=1}^{M} H_i$, and E_1 is the total variance in the data set $E_1 = \sum_i \|\mathbf{x}_i - \overline{\mathbf{x}}\|^2$, where $\overline{\mathbf{x}}$ is the centroid of the data. Thus, the values are (in most cases) between [0,1]. The value of H is 6.4 for the system map and 6.6 for the mills map, both with $\sigma = 1$. The total variance is 16742 for the system map, and 248040 for the mills map. The errors for k-means have been scaled by E_1.

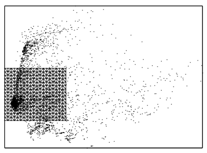
	system				**mills**			
t	0	t_1	t_1+t_2	k-means	0	t_1	t_1+t_2	k-means
E_d/HE_1	0.346	0.137	0.144		0.753	0.547	0.535	
E_{qx}/HE_1	0.109	0.043	0.035	0.030	0.516	0.206	0.188	0.145
E_{nb}/HE_1	0.219	0.046	0.053		0.235	0.254	0.245	
E_{nv}/HE_1	0.018	0.047	0.057		0.002	0.088	0.101	

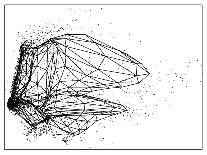

(a) System: $t = 0$

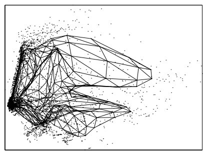

(b) $t = t_1$


(c) $t = t_1 + t_2$


(d) k-means


(e) Mills: $t = 0$


(f) $t = t_1$


(g) $t = t_1 + t_2$

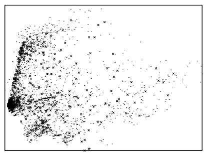

(h) k-means

Figure 3.6: PCA-projections of the data (the gray dots) and the prototypes (the mesh) at different phases of training (a-c) and (e-g). Figures (d) and (h) show the results of k-means algorithm.

An important group of variants is based on replacing the prototype vectors with some more complicated models. To do this, the winner search and update definitions in the basic SOM must be replaced. An example of this based on generalized median is presented in Section 3.2.1. Other variants along this theme include maps of auto-regressive models [94], linear models [59], subspaces [83], and probabilistic models [52].

3.2.1 Generalized median SOM

Recently, a new version of the Batch map was introduced which expands the applicability of the SOM to data domains where the objects are not fixed-length vectors, for example strings, aminoacid or phoneme sequences [85]. In this algorithm, new prototype vectors are calculated using generalized median or generalized mean, which can be determined for any set of objects for which a pairwise distance function exists. The generalized median $\bar{x}_j$ of a set of objects X_j is:

$$\bar{x}_j = \arg\min_{\bar{x}} \sum_{x_j \in X_j} d(x_j, \bar{x}) \tag{3.28}$$

where $d(\cdot)$ is some distance measure defined for all pairs of objects $(x_j, \bar{x})$. Likewise, the generalized mean is:

$$\bar{x}_j = \arg\min_{\bar{x}} \sum_{x_j \in X_j} d(x_j, \bar{x})^2 \tag{3.29}$$

Note that in either case $\bar{x}$ need not be part of the original set of objects X_j.

The modified Batch map algorithm using generalized mean (or median) is:

1. *Search for the BMUs.* For each map unit j, collect the set V_j of objects x_i that are nearest to the prototype $\bar{x}_j$ of that map unit.

2. *Update prototypes.* For each map unit j, construct the union of object sets of the map unit j and its neighboring map units $\mathcal{N}_j$ (see Figure 3.1):

$$V_{\mathcal{N}_j} = \bigcup_{k \in \mathcal{N}_j} V_k$$

   Let the new prototype to be the generalized median (or mean) of the set $V_{\mathcal{N}_j}$.

3. Repeat from step 1 until the algorithm has converged, or a predefined number of iterations is reached.

3.2.2 Soft topographic vector quantization

Some illustrative ideas of the SOM are present in a family of algorithms called soft topographic vector quantizers (STVQ) [98, 38]. In these algorithms, the vector quantization problem of reconstructing the original data set is complicated by unreliable transmission of the best-matching unit index, see Figure 3.7. The topographic organization of the quantization prototypes is used to minize the error due to errors in the transmission. The transmission error probabilities are used as neighborhood function values $h_{ij} = P(i \rightarrow j)$ between different prototype indexes.

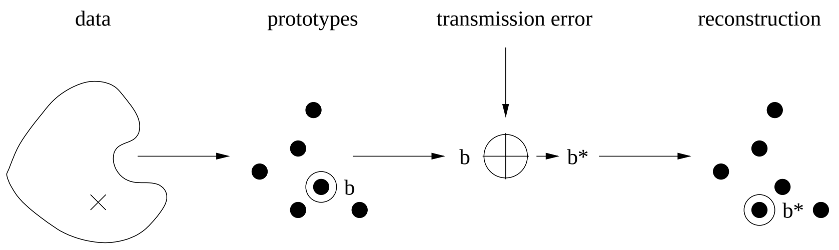


Figure 3.7: The error model in soft topographic vector quantization.

The optimization is done using Expectation-Maximization (EM) (see for example [8]). In expectation step the probabilities of data points to be assigned in each unit j are calculated:

$$P(j\,|\,\mathbf{x}_i) = \frac{e^{-\frac{\beta}{2}\sum_k h_{jk}||\mathbf{x}_i-\mathbf{m}_k||^2}}{\sum_l e^{-\frac{\beta}{2}\sum_k h_{lk}||\mathbf{x}_i-\mathbf{m}_k||^2}},$$

where β is a parameter corresponding to inverse of noise. In the maximization step, new prototypes are calculated:

$$\mathbf{m}_j = \frac{\sum_i \mathbf{x}_i \sum_k h_{jk} P(k\,|\,\mathbf{x}_i)}{\sum_i \sum_k h_{jk} P(k\,|\,\mathbf{x}_i)}.$$

In case of crisp winner assignment ($\beta \to \infty$), the expectation step becomes a winner-take-all approach $P(j\,|\,\mathbf{x}_i) = \delta(j, b_i)$, where $\delta(\cdot,\cdot)$ is the Dirac delta function, and $b_i = \arg\min_j \sum_k h_{jk}||\mathbf{x}_i - \mathbf{m}_k||^2$ (Eq. 3.7). If the winner definition is further replaced with $b_i = \arg\min_j ||\mathbf{x}_i - \mathbf{m}_j||^2$ (Eq. 3.1), the Batch map algorithm results. Thus, the Batch map learning rule can be regarded as a computationally efficient approximation of STVQ.

3.2.3 Related algorithms

k-means. Possibly the closest well-known relative of the SOM is the *k*-means vector quantization algorithm [107] (see Section 3.1.3). A closely related method is the fuzzy *c*-means algorithm [6], which differs from *k*-means in that each data sample can belong to a number of clusters, even to all of them, in varying degrees. The minimized error function is:

$$E_{fcm} = \sum_{i=1}^{N}\sum_{j=1}^{N} (\mu_j(\mathbf{x}_i))^b ||\mathbf{x}_i - \mathbf{m}_j||^2, \tag{3.30}$$

where b is a parameter controlling the fuzziness of the model, and $\mu_j(\mathbf{x}_i)$ is the fuzzy membership of data sample $\mathbf{x}_i$ is cluster j. It is usually constrained by $\sum_j (\mu_j(\mathbf{x}_i))^b = 1, \forall i$ and calculated based on input space distances $||\mathbf{x}_i - \mathbf{m}_j||$. In contrast, in the SOM the corresponding coefficients are the neighborhood function values which are defined in the output space.

A more principled manner to assign non-crisp memberships is to approximate the data with Gaussian Mixture Models (GMM) (see for example [8]). The underlying assumption is that the data comes from a mixture of Gaussian distributions. A number of such distributions are spread into the data and their parameters (centers and covariance matrices) are optimized using EM.

Principal curves and surfaces represent a concept close to the SOM. In principal curves, the idea is to find the central curve (or surface) going through the data manifold [46]. Each point on the principal curve is the average of all points that project to it. The SOM prototype vectors are conditional averages of the data, and thus it can be regarded as a discrete counterpart of the principal curve [18, 115].

VQ-P. The SOM is a combined vector quantization and vector projection algorithm. There are also other algorithms which combine properties from vector quantization and vector projection, for example Vector Quantization and Projection neural network [24], and combinations of any vector quantization and vector projection algorithms, like *k*-means and Sammon's mapping [33] or *k*-means and MDS [120].

Generative Topographic Mapping (GTM) was proposed by Bishop *et al.* [9] as an alternative to the SOM. It maps a low-dimensional latent (or output) space into the data (or input) space using some mapping function $f(\mathbf{r};\mathbf{W})$ where $\mathbf{r}$ is some point in the latent space, and $\mathbf{W}$ is a matrix of parameters for the mapping function. The mapping is optimized through maximizing the log-likelihood of the training data:

$$\mathcal{L}(\mathbf{W},\beta) = \sum_{i=1}^{N} \ln\{\frac{1}{M}\sum_{j=1}^{M} p(\mathbf{x}_i\,|\,\mathbf{r}_j;\mathbf{W},\beta)\} \tag{3.31}$$

$$p(\mathbf{x}_i\,|\,\mathbf{r}_j;\mathbf{W},\beta) = (\frac{\beta}{2\pi})^{d/2} e^{-\frac{\beta}{2}\|f(\mathbf{r}_j;\mathbf{W})-\mathbf{x}_i\|^2} \tag{3.32}$$

where β is the inverse of the variance of noise. The $\mathbf{r}_j$ are a set of M lattice points in the latent space which determine the points $\mathbf{m}_j = f(\mathbf{r}_j;\mathbf{W})$ in the data space where the probability density is estimated[2]. They correspond to the map units in the SOM.

The GTM is, in essence, a constrained mixture of Gaussians in which the model parameters $\mathbf{W}$ and β are estimated using EM algorithm. In [9], the mapping function f is a generalized linear regression

$$f(\mathbf{r};\mathbf{W}) = \mathbf{W}\Phi(\mathbf{r}) = \sum_i \phi_i(\mathbf{r})\mathbf{w}_i, \tag{3.33}$$

where $\phi_i(\mathbf{r})$ are a set of Gaussian kernels situated in the latent space, and $\mathbf{w}_i$ are their corresponding locations in the data space. In the E-step, the responsibility of each lattice point $\mathbf{r}_j$ for each data point $\mathbf{x}_i$ is calculated as:

$$r_{ji} = \frac{p(\mathbf{x}_i\,|\,\mathbf{r}_j;\mathbf{W},\beta)}{\sum_k p(\mathbf{x}_i\,|\,\mathbf{r}_k;\mathbf{W},\beta)}. \tag{3.34}$$

In the M-step, new estimates for the model parameters $\mathbf{W}$ and β are found (see [9] for details).

S-Map was proposed by Kiviluoto and Oja [74] as an alternative to GTM. It is a crossbreed between SOM and GTM. Like in SOM, each map unit j has a fixed place $\mathbf{r}_j$ on the map grid and an associated prototype vector $\mathbf{m}_j$. Each map unit generates a Gaussian distribution in the data space with center at the map unit prototype, and with variance $1/\beta$. Like in GTM, the softmax responsibilities:

$$r_{ji} = \frac{e^{-\frac{\beta}{2}\|\mathbf{m}_j-\mathbf{x}_i\|^2}}{\sum_{k=1}^{M} e^{-\frac{\beta}{2}\|\mathbf{m}_k-\mathbf{x}_i\|^2}}$$

[2]The notation has been changed to match the convention used in this thesis.

are used in the update step. The learning rule of SOM (Eq. 3.2) is changed to:

$$\mathbf{m}_j = \mathbf{m}_j + \alpha \left(\sum_{k=1}^{M} h_{b_i j} r_{ki} \right) [\mathbf{x}_i - \mathbf{m}_j].$$

The simulations in [74] suggest that S-Map is better at self-organization than GTM. As opposed to SOM, the S-Map has an inherently probabilistic nature in that it generates a probability density function in the data space. In Publication 4, this was compared with SOM-based density estimation.

3.3 Discussion

The SOM simultaneously quantizes the data with a representative set of prototype vectors, and orders — or projects — the prototype vectors on a regular map grid. The relative importance of the quantization and projection tasks is governed by the size of local neighborhood — in effect, the neighborhood radius. With small neighborhood radius, the SOM approaches the unordered set of prototype vectors produced by *k*-means algorithm. With large radius, the neighborhoods order the map prototypes on a tightly packed grid. The desired result is usually somewhere between these extremes.

When utilizing the SOM for data analysis, it is important to know which variables and properties are well represented by the map. Because current methods do not offer an obvious way to fix these beforehand, the data miner may need to iterate a bit between training SOMs with different parameterizations, and validating them with different kinds of quality measures.

- The elements of the distortion measure discussed in Section 3.1.2 can be used to compare the quantization and projection properties of the SOM to each other. In addition, the quantization error E_{qx} can be compared directly to the error of the *k*-means algorithm to obtain an idea of the quantization quality of the SOM. Unfortunately, the same cannot be done to the neighborhood related errors. Some other measure must be used to quantify the topological quality of the SOM, for example neighborhood preservation or trustworthiness [136].

- Publication 8 proposes two novel measures for quantifying the "viewpoint" of a given quantization: how well each variable is quantized, and how much resources (in effect, quantization points) does it take to quantize each variable. Such measures are important in the investigated data exploration process because the quantization forms a basis for many other algorithms. However, besides knowing how well each variable is represented in the quantization, the user generally wants to have control over their importances. Unfortunately, the proposed methods only provide feedback for this: they cannot be used to directly determine good scaling factors for the variables.

The SOM has a number of variants and related algorithms. Sometimes they are much better suited for a specific data mining task than the basic SOM. Although this thesis concentrates specifically on the SOM, it is important to realize that the investigated data exploration process framework is not dependent on the SOM. If the needs or personal preferences of the user so require, the SOM can be easily replaced with other methods as long as they possess a few key characteristics:

- The data are represented by prototypes.

- The prototypes are connected by neighborhood relations.
- For visualization purposes, the prototypes must have positions on a low-dimensional space which reflect their distances or ordering in the high-dimensional space.

Chapter 4

Data exploration methods

In this chapter, three important data exploration tasks are discussed: visualization, cluster analysis and modeling. For the two first tasks, overviews of the corresponding problem domains are given, and the investigated SOM-based methods are introduced and compared to other methods with similar aims or properties. For the last task, local modeling based on SOM is shortly introduced.

Visualization of multivariate data is discussed in Section 4.1. The goal of visualization in general is to convey large amounts of detailed information about the data to the data miner. To be efficient, the visualizations must be easily understandable and take the strengths of the human visual system into account. An overview of the state of the art in information visualization is given by Ware [147]. Somewhat older, but still an excellent account of conventional visualization techniques is given by Tufte [129].

Cluster analysis, discussed in Section 4.3, is a widely researched topic in data analysis [28, 71]. Its goal is to partition the data into natural groups. In data exploration, however, the groups themselves are not sufficient. In order to provide insight to the data, it is also important to describe these groups: what properties are typical for the objects in the groups, and what makes them different from objects in the other groups.

Visualization and clustering are the main applications of SOM in data analysis. A particular strength of SOM is that it answers the needs of both tasks within a common framework. For all three tasks, the SOM provides an initial organization of the data. The actual visualizations, clusterings or models are usually acquired after some post-processing. How this post-processing should be done is not quite obvious. One of the main contributions of this thesis is to investigate these post-processing methods, to enhance them and to develop new methods to answer the needs of the tasks better. Section 4.2 presents an overview of SOM-based visualization methods. Section 4.3.3 discusses SOM-based clustering techniques and Section 4.3.5 how the clusters can be characterized. In Section 4.4 construction of local models based on SOM is introduced. In Chapter 5, these methods are integrated into the data exploration process.

4.1 Visualization of multivariate numerical data

Visualization is a fundamental methodology in exploratory data analysis for several reasons. Looking at the data allows the data miner to speculate about its properties based on his/her own intuition and domain knowledge. Visualization also makes it possible to compare raw data to constructed models, and find unexpected details and errors

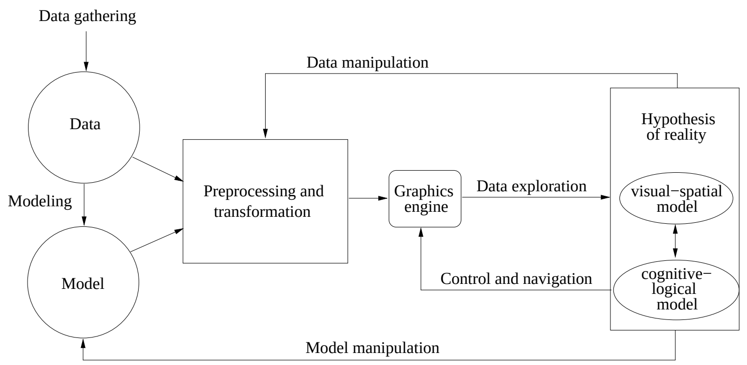


Figure 4.1: An interactive visual data exploration environment. Based on visualizations produced of the preprocessed data, the viewer forms a hypothesis, or mental model, of the properties of the data. The hypothesis is tested and refined interactively by manipulating the visualization and the data, and by comparing the data to the associated computational models. This diagram is based on Chapter 10 in [147].

in the data or in the models [129]. This is important since outliers or unexpected shapes of the data manifold can easily corrupt the results of analysis methods. Figure 4.1 presents an interactive visualization environment, where visualization parameters are adjustable online, and the user can easily zoom in on interesting details.

Multivariate numerical data is usually visualized using scatterplots. Scatterplots consist of a large number of markers distributed in a low-dimensional space. They are used to detect

1. groups (clusters) of similar objects, and
2. relationships between variables.

Using only physical coordinates, at most 3-dimensional data can be shown[1]. To increase the number of visualized variables, three basic techniques are outlined below: (1) reducing dimensionality of the data set by projection techniques, (2) using parameterized visual markers to encode different variables, and (3) using multiple visualizations simultaneously by linking them together.

4.1.1 Projection

Vector projection was discussed in Section 3.1.4. Projection methods try to find low-dimensional coordinates that preserve the distances (or the order of distances) between the originally high-dimensional objects. Of course, it is usually impossible to make an exact reproduction. Reliable projections can be achieved if the variables are highly correlated, and thus contain a lot of redundant information, or if the data contains a lot of noise which can be discarded.

The projection is a lossy similarity encoding between data objects. From the projection plot, one can see the clusters in the data, as well as the general "shape" of the

[1]Presentation media — a paper or a computer screen — typically limits the visualizations to 2D. In interactive visualization environments also 3D-visualizations can be used efficiently because the viewer can rotate and manipulate the visualizations.

data cloud[2]. However, relations between individual variables are lost since the new coordinates are complex linear or non-linear combinations of the original variables. Thus, projection visualizations are only applicable for detection of similar groups of objects. Other techniques must be used to detect relationships between variables.

Projection techniques

The classical projection method is multi-dimensional scaling (MDS) [89] which tries to preserve pairwise distances between all objects while reducing the dimensionality as defined in Eq. 3.24 on page 19. The best possible linear solution for this projection problem is given by principal component analysis (PCA), see for example Chapter 3 in [5]. In PCA, the directions are found which account for most of the variance in the data. This is done by calculating the eigenvectors $\mathbf{e}_1, ..., \mathbf{e}_d$ and corresponding eigenvalues $\lambda_1, ..., \lambda_d$ of the covariance matrix of the data, and ordering them by decreasing eigenvalues $\lambda_1 > \lambda_2 > ... > \lambda_d$. The first direction $\mathbf{e}_1$ accounts for most — specifically $\frac{\lambda_1}{\sum_i \lambda_i} 100\%$ — of the variance in the data, the second for the second largest amount, and so on. By projecting data to the space spanned by the first few eigenvectors as much of the variance is preserved as possible. The sum of the corresponding eigenvalues gives the amount of variance preserved in the projection, and thus indicates the error made in the low-dimensional projection. For example, a projection to a 2-dimensional plane is defined as:

$$\mathbf{y} = \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{bmatrix} \mathbf{x}. \tag{4.1}$$

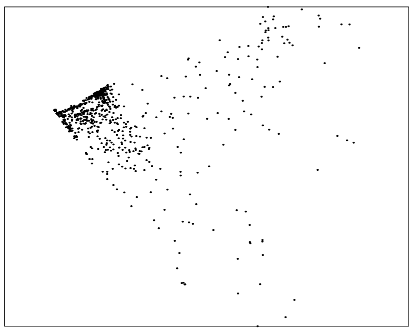
Examples of PCA-projections are shown in Figure 4.2.

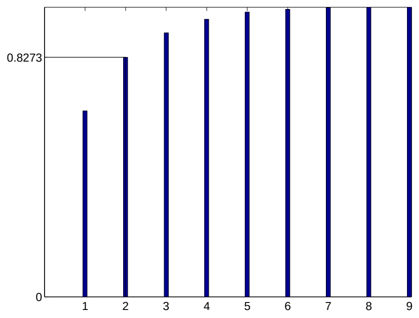
The PCA-projection is a special case of projection pursuit techniques, which maximize some kind of performance or interestingness index over the possible projections, see for example [18]. For PCA this index is variance, while for many projection pursuit techniques it is some measure of non-gaussianity. For example, in independent component analysis (ICA) [53], the objective is to find projection directions where the marginal distributions of projected points are independent of each other. Notice that the performance index may not have anything to do with distance preservation. When interpreting visualizations based on projection pursuit techniques, the objective of the performance index needs to be taken into account.

Like MDS, the PCA and ICA techniques discussed above are globally tuned techniques for which large distances are more important than small details, as discussed in Section 3.1.4. The techniques are also linear, and therefore may fail when other, more flexible projection techniques might succeed. In addition to Sammon's mapping

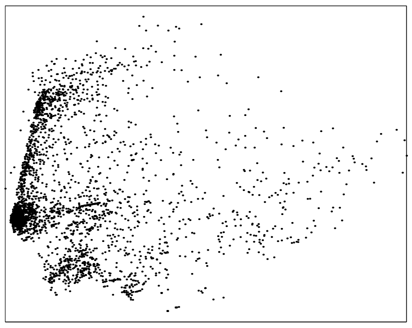
[2] Any clusters found from these visualizations are subject to the interpretation of the observer. Research in cognitive psychology has produced a set of so-called Gestalt laws ([75, 147]) which describe the way the human visual system detects patterns in images. Items which have some of the following six properties tend to be grouped together:

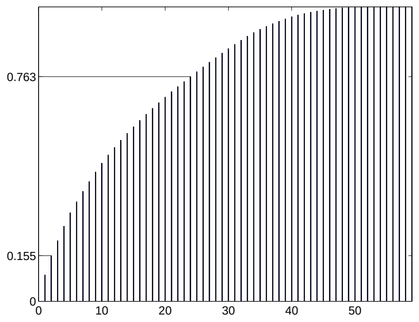
1. Items close to each other.
2. Items which are similar to each other, for example of the same size, shape or color.
3. Items in a region with similar density of items.
4. Symmetrically positioned items.
5. Items which are within a closure.
6. Items which are connected, or when combined would produce a smooth continuity.


(a) System: PCA projection



(b) System: scree-plot


(c) Mills: PCA projection



(d) Mills: scree-plot

Figure 4.2: PCA-projections of the system (a) and mills (c) data sets. The corresponding scree plots (b and d) show the cumulative sum of eigenvalues. They give some indication of the intrinsic dimensionality of the data set. For the system data set 83% of the total variance is preserved in the 2-dimensional projection. For the mills data set only 15% is preserved.

and CCA introduced earlier, there are a number of such techniques, see for example [26, 87, 97, 100, 118, 121, 127]. The SOM as a projection algorithm [64] preserves local neighborhood sets. In certain situations this kind of projection is very useful, see Section 4.2.3.

Projection into color space

Besides spatial position, also other scatter plot marker properties can be used to indicated similarity, for example color: data objects close to each other are assigned similar colors. While the resulting similarity encoding is not as accurate as the one produced by spatial projection, it is useful for linking multiple visualizations together [49, 68, 142], or when the position information is needed for other purposes.

A color coding can be constructed by defining a smooth coloring in a low-dimensional manifold, and projecting the data onto this manifold, for example as follows:

1. A 1-dimensional SOM is trained using the data. The topology of the map can be either circular or a simple chain. The trained SOM forms a principal curve going through the data manifold.

2. A color from the color hue circle (from the HSV color model, see for example [147], with hue = ϕ, saturation = 1 and value = 1) is assigned to each map unit i of the 1-dimensional SOM. The colors can be assigned equidistant from each other $\phi_i = 2\pi i/M$ or the distances between neighboring prototypes can be taken into account: $\phi_i = 2\pi \sum_{j=1}^{i} \|\mathbf{m}_{i+1} - \mathbf{m}_i\| / \sum_{j=1}^{M-1} \|\mathbf{m}_{i+1} - \mathbf{m}_i\|$.

3. Each data point picks the same color as its BMU.

A similar, although a bit more complicated approach, was proposed in [68]. In order to create reliable color codings, the smoothness of the coloring on the projection manifold is very important. In [69] a color projection algorithm is proposed which takes perceptual differences between colors into account to build as faithful representations as possible.

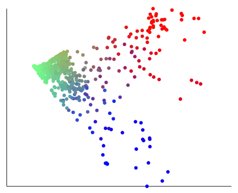
In Figure 4.3 two color projections are shown for the example data sets. The ones on the right have been made using the algorithm above. The ones on the left have been done using the algorithm proposed by Himberg [50]. Although the coloring on the left has preserved the original distance information better, the coloring on the right has more granularity and is therefore used in the rest of this thesis.

4.1.2 Parameterized markers

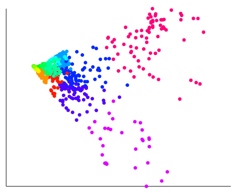
Instead of using the same kind of visual marker for each data object in the scatterplot, one can modify the properties — for example size, color or shape — of each marker. The spatial coordinates are still used to encode the most important variables, but in addition a few more can be shown using the parameterized properties.

Glyphs

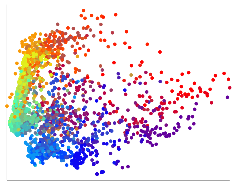
Complicated shapes, or glyphs, are one way to incorporate additional information into visualizations. Examples of glyphs include stick figures, fan and star plots (see for example [147]). A single fan plot marker consists of several rays extending from a common starting point, see Figure 4.6a on page 39. While the number of rays can be made quite large to encode very high-dimensional objects, it soon becomes rather



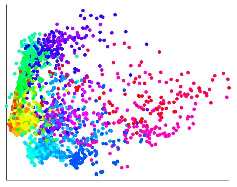
(a) System



(b) System



(c) Mills



(d) Mills

Figure 4.3: Color codings of the data sets shown together with their PCA-projections. The colorings in (a) and (c) have been done using the algorithm proposed by Himberg [50]. The colorings in (b) and (d) have been done by a projection of data on the color circle, as described in Section 4.1.1. Just cluster centroids were used in the projection and thus all points in a cluster have been assigned the same color.

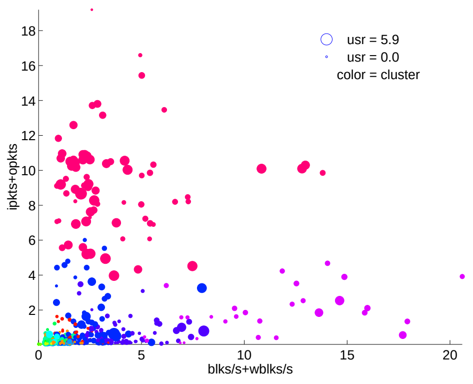


Figure 4.4: A scatterplot of five variables of the system data (see Appendix A). Both spatial coordinates are sums of two variables (`wblck/s` + `blck/s` and `ipkts` + `opkts`), the size of each marker indicates the value of `usr`, and color indicates the cluster the sample belongs to.

difficult to identify which variable a certain ray corresponds to. The technique works best when the variables have a natural ordering, for example frequencies in a spectrum.

A famous example of glyphs are the Chernoff's faces [19]. The marker for each data object is a face. The variables control different features of the face: lengths, positions, curvatures and angles of nose, eyes, mouth, ears and chin, see Figure 4.6b. While this is an intriguing idea, it is not very useful in practice because of several reasons:

- Except for certain pairs the features are not easily comparable with each other. This makes it hard to compare the relative closeness of two pairs of faces which are different in different ways.
- There is a hierarchy in the importance of features which is hard to quantify. The human observer pays much more attention to eyes than, say, ears.
- The faces require a considerable amount of actual space in the visualization, so plotting a large number of them is not feasible.

Obviously, many complex glyphs suffer from similar weaknesses. However, although glyphs are poor at indicating the magnitude of difference between data samples, they can embed a lot of information and allow the user to compare individual data samples efficiently.

Pre-attentive features

For efficient visual interpretation, the properties used for encoding should correspond to the strengths of the human visual system. Studies in cognitive psychology have shown that there are certain visual features that the human visual system analyzes very fast, pre-attentively [47, 147]. These pre-attentive features are processed from the whole field of view in parallel, and thus interpretation happens at a glance. The most important pre-attentive features are:

- spatial position: both 2D-position and depth
- shapes: size, curvature, spatial grouping, added marks, numerosity
- line or texture shapes: line orientation, length, width, and collinearity
- color: hue and intensity
- motion

Using combinations of such features, one can encode several variables simultaneously. Unfortunately, several of the features interfere with each other. There are only a few combinations of these features which retain the pre-attentive processing property with respect to all features. Spatial position is a particularly strong feature, and it can be efficiently combined with, for example, shapes or colors. Figure 4.4 shows an example of encoding with different visual features.

4.1.3 Multiple small visualizations

The third elementary solution is to break the visualization into a number of smaller ones. In [129], these are called small multiples. The small multiples are linked together so that one can immediately identify the same object (or at least the same group of objects) from the different visualizations [15]. Scanning through the small multiples is very efficient because they are interpreted in a similar way. Since each small multiple can be investigated independently of the others, a lot of detailed information can be incorporated while still retaining the ability to compare the small multiples with each other.

The linking between multiples can be done in several ways, some of which are used in Figure 4.5.

- Linking can be done by *position*. In this case, the position of each object remains fixed in each small multiple. Other marker properties, such as color or size, are used to indicate the values of the visualized variables. Linking by position is extensively utilized in visualization of SOM (see Section 4.2).

  Examples of linking by position are shown in Figure 4.5. In Figure 4.5a, the position derived from PCA-projection is used in each small multiple. Using projection for linking has the advantage that the position acts not only as a link between small multiples but also as a similarity encoding. Another simple example of linking by position is a multiaxis time-series plot, as shown in Figure 4.5b. In that case, only position on the x-axis remains fixed for each object.

- Linking by *color* is less accurate than linking by position[3]. However, linking by color is very useful when it is sufficient to clearly distinguish just groups of objects from each other, instead of individual objects. Good color codings can be constructed, for example, using the color projection techniques mentioned earlier.

  Color coding can be used, for example, to enhance the scatterplot matrix visualization technique. A scatterplot matrix consists of pairwise scatterplots between each pair of the variables. It is used to quickly find relationships between variables. An obvious deficiency of the technique is that the number of scatterplots grows quadratively with the number of variables. Another deficiency is that it is very hard to identify the same item from all scatterplots. However, this can be rectified somewhat by using color to link the scatterplots together, as in Figure 4.5c.

- A very strong linking technique is to connect objects *explicitly using lines*, as done in the parallel coordinates visualization technique. Parallel coordinates consist of a number of parallel axes, each of which shows the value of one variable [55]. Linking is done by drawing lines from each successive axis to the next, according to the value of the object for that variable. An example of parallel coordinates visualization is shown in Figure 4.5d.

  A problem with this technique is that the lines get mixed very easily, and thus it is hard to see which line segment corresponds to which object. In Figure 4.5d, linking with color is used to indicate groups of similar objects. In interactive environments, brushing techniques [4] can be used: by brushing with the mouse the user selects a set of objects and these objects are emphasized in all visualizations.

- In [26], *motion* is used to link different projections of a data set together. Rather than use several small multiples side-by-side, the projections are shown one after

[3]While the human visual system can easily distinguish between millions of different colors, it requires considerable effort to find two points with exactly the same color when there are a large number of different colors. According to Healey [47], the maximum number of colors that can be separated from each other pre-attentively is about seven.

the other. The transition from one projection to the next is done in small steps so that the points appear to move smoothly from one place to another.

4.2 Visualization based on SOM

The SOM-based visualization is investigated in Publication 3. Its contribution is to provide an overview, categorize the SOM-based techniques, and relate them to other visualization techniques. In addition, a few enhancements are proposed which are further studied in Publications 4 and 5.

In the visualizations, the SOM acts in two roles. In cluster and variable visualization techniques the prototypes of the SOM are regarded as a representative sample of the data: the original data is replaced with a smaller set where the effect of noise and outliers is decreased. It is assumed that the properties seen from visualizations of the prototypes will also hold for the original data. For this reason, some caution is in order before making any far-reaching conclusions based on the SOM visualizations.

Other techniques regard the SOM as a model of the data, and they compare the data to this model, or different data samples or sets to each other in the context provided by the model. For example, the map grid can be used to compare data samples to each other by comparing their locations on the map grid.

4.2.1 Map grid as a visualization platform

The SOM forms a low-dimensional map of the data. The map grid is an ordered representation of the data: neighboring regions on the map are similar to each other, while regions far from each other are different from each other (unless the map has folded badly). The map grid has the following properties:

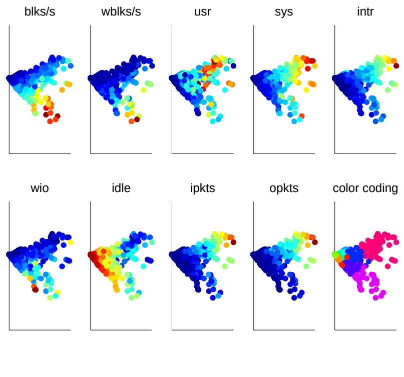
- The shape of the map grid is predefined so that each map unit has a unique place and equal size. Therefore graphs and other kinds of complicated glyphs can be easily inserted into each map unit without fear of overlaps. Figure 4.6 shows several examples of glyphs visualizing the prototype vectors in the SOM of the system data.

- Since the density of map prototypes follows roughly the density of the data, the map grid provides a kind of automatic focus following the data density. Compare, for example, coloring visualizations in Figure 4.7. The color coding is hardly visible at all in dense areas of the PCA-projection, whereas the SOM shows all areas equally well.

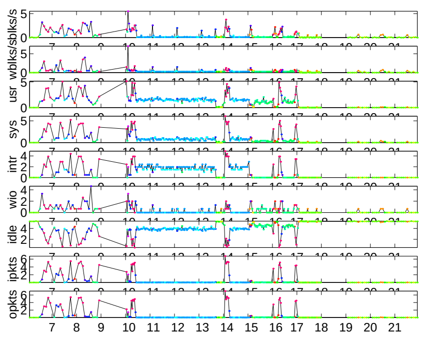
- The map preserves topological neighborhoods, but relative distances in the original space are not well represented.

- The map grid forms a highly non-linear 2-dimensional manifold in the original space. Thus the coordinate axes of the output space do not have any clear interpretation in terms of the original variables.

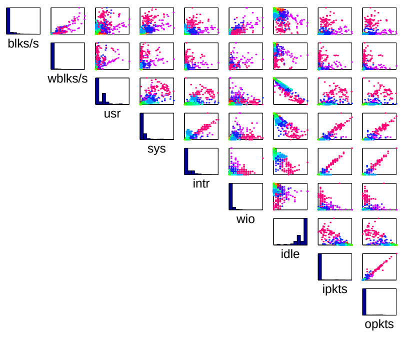
Each map unit has a set of associated property values, for example the values of variables in the prototype vectors. Usual SOM visualizations show these values in all map units thus allowing the comparison of different map units and, since neighboring



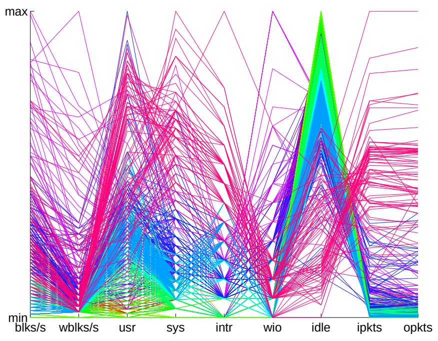
(a) Component planes



(b) Time-series



(c) Scatterplot matrix



(d) Parallel axis

Figure 4.5: Small multiples of the system data set. In the component planes figure (a), linking is done by position. Each subplot corresponds to one variable, and each dot in each subplot to one data sample. The coordinates are from the PCA-projection of the data. The name “component planes” comes from a similar technique used in visualization of SOM, see Figure 4.8. The last component plane shows the color coding of the map, and thus links the component planes with (b-d). In the time-series plot (b) the system data for Tuesday 6:00 to 22:00 is depicted. The samples have been colored using the color coding. The scatterplot matrix (c) shows the pairwise scatterplots of all variable pairs. The objects in the scatter plots have been linked together using color. On the diagonal, the histograms of individual variables are shown. In the parallel coordinates visualization (d) each vertical axis corresponds to one variable. Each horizontal line corresponds to one object such that linking is done explicitly using lines. The line color encodes similarity between objects. Figures (a), (c) and (d) all show all of the data, and can be used to detect correlations between variables.

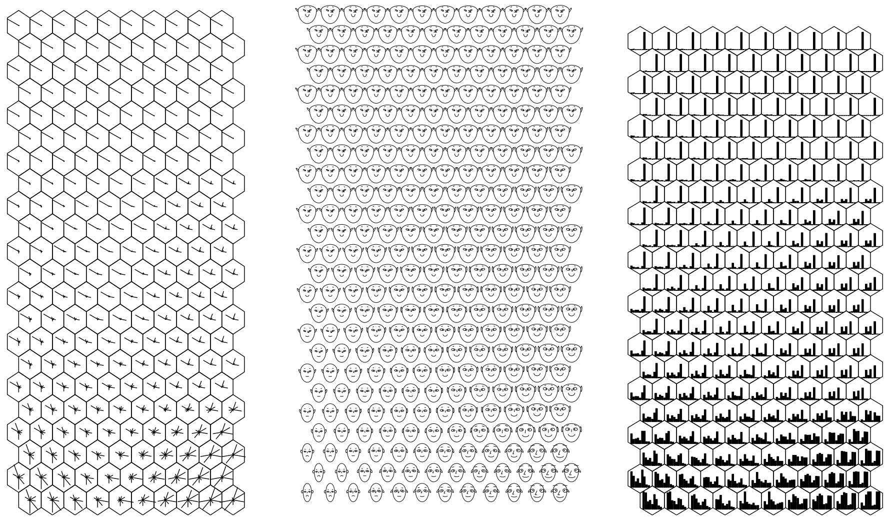

(a) Fan plots (b) Chernoff's faces (c) Bar charts

Figure 4.6: Visualizations of the prototype vectors in the SOM of the system data. The 9-dimensional prototype vector in each map unit has been plotted as a glyph on the map grid: (a) fan plots, (b) Chernoff's faces, and (c) bar plots. In (a) and (c), the hexagonal map units borders are also shown.

map units will typically have similar values, of map regions with respect to the shown properties.

Often, several map grids each indicating the values of a different property are shown side by side. Usually, the map grid coordinates are used to link the visualizations together, as in Figure 4.6. However, sometimes physical coordinates cannot be used for linking purposes, for example when plotting variables against each other, as in a scatterplot matrix. In such a case color codings can be used [49, 68], as in Figure 4.7. To emphasize neighborhood relations, also lines can be drawn between neighboring map unit markers.

4.2.2 Visualization of clusters

Techniques to visualize the shape and cluster structure of a data cloud are usually based on vector projections. Since the shape of the SOM grid is predefined, it is not useful for this as such. Instead, the map prototype vectors must be projected on a low dimensional space using some other projection technique. Besides physical coordinates, also color coding techniques have been used for this purpose [50, 68, 69].

However, the most commonly used technique to visualize the clusters on the SOM are distance matrices. In these techniques, the distances between each unit i and the units in its neighborhood $\mathcal{N}_i$ are calculated:

$$\mathcal{D}_i = \{\|\mathbf{m}_i - \mathbf{m}_j\| \mid j \in \mathcal{N}_i, \, j \neq i\}. \tag{4.2}$$

The distances, or for example the median of these distances [88], for each map unit are typically visualized using color, although also other techniques are possible [41, 54, 103]. The unified distance matrix (U-matrix) [134] visualizes all distances between each map unit and its neighbors. This is possible due to the regular structure of the map grid: it is easy to position a single visual marker between a map unit and each of its neighbors. Since the map prototypes follow the probability density function of the data, the neighbor-distances $\mathcal{D}_i$ are inversely proportional to the density of the data. Thus, cluster borders can be identified as "mountains" of high distances separating "valleys" of low distances. This interpretation can also be used in clustering, see Section 4.3.3. Figure 4.7 presents examples of both spatial and color projections as well as the U-matrix visualizations of the system and mills maps. Several clusters can be seen, especially from the U-matrices.

4.2.3 Visualization of variables

Component planes

To visualize variables using the SOM, a technique called component planes is used. For each visualized variable, or vector component, one SOM grid is visualized such that the colors (or for example sizes) of the map unit markers change according to the visualized values, see Figures 4.8 and 4.9. Relationships between variables can be seen as similar patterns in identical places on the component planes: whenever the values of one variable change, the other variable changes, too [139].

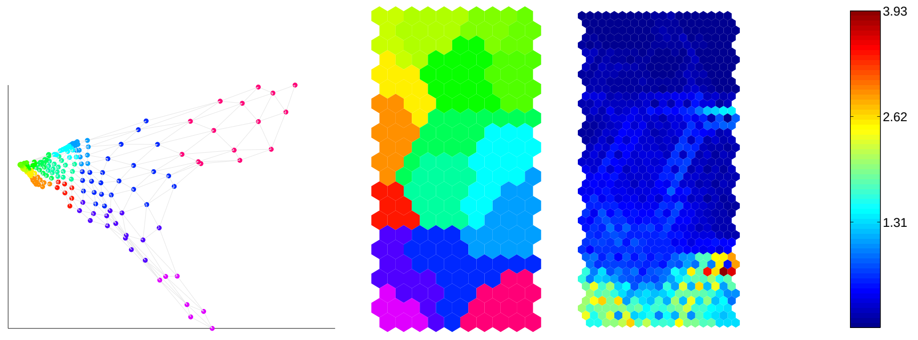
Although any kind of projection could be used to link the component planes together, the SOM grid works particularly well in this task. Because of the dynamic focus of the map, the behavior of the data can be seen irrespective of the local scale. Compare, for example, the component planes visualizations in Figures 4.5a and 4.8. Although the PCA projection in Figure 4.5a reflects the shape of the data cloud better, it is much harder to see the patterns in it because there are gaps and many of the markers obscure each other.

However, as pointed out in [92], in some cases the visual impression of dependency may be wrong. Therefore, it is important to validate the found dependencies with other methods, for example scatterplots. On the other hand, the SOM also reduces the effect of noise and outliers, and thus may actually make any dependencies more clear than they are in the original data, as can be seen from Figure 4.10.

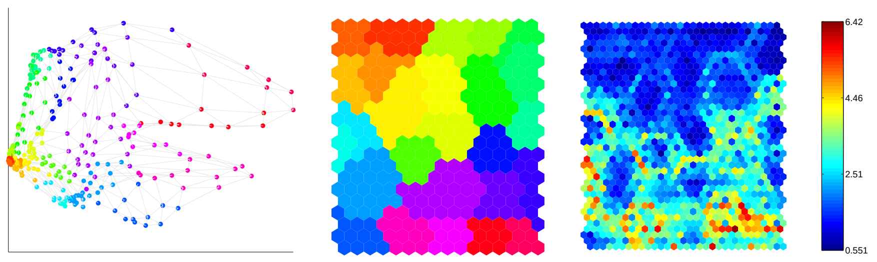
Clustering variables

When a scatterplot matrix (Figure 4.5c) is used for visualization of relationships between pairs of variables, one has to scan through $(d-1)(d-2)/2$ plots. In case of component planes, there are only d plots, but one has to scan through pairs of plots, so the complexity of scanning is not really any smaller. However, it can be aided by organizing the component planes such that those corresponding to correlated variables are positioned near each other. Figure 4.9 shows an example of organizing component planes in this manner.

The organization of the component planes is based on measuring dependencies between variables, collecting these measurements into vectors $\mathbf{v}_i$, and then grouping similarly dependent variables. In Publication 5, four different ways to measure the dependencies are compared. The way that gave best results is based on correlation



(a) System: PCA (b) Coloring (c) U-matrix



(d) Mills: PCA (e) Coloring (f) U-matrix

Figure 4.7: Cluster visualizations of the SOMs of the system (a, b, c) and mills (d, e, f) data sets. PCA-projection (a, d), color coding (a, b, d, e), and the U-matrix (c, f). In (a) and (d), the map unit coordinates come from a PCA-projection. In (b, c) and (e, f), the map grid coordinates are used. The PCA-projection (a, d) shows the shape of the data set better than the SOM grid, but it is hard to see local details in the dense areas. In interactive visualization environments, this is not a big problem because the user can zoom in on interesting details. However, in static visualizations such as above this is not possible. As opposed to PCA-projection, the map grid has equal amount of space for each map unit, and thus map units even in the dense areas can be seen clearly.

coefficients between variables:

$$\mathbf{v}_i = [|c_{i1}|, |c_{i2}|, ..., |c_{id}|] \tag{4.3}$$

$$c_{ij} = \frac{1}{\sigma_i \sigma_j} \sum_{k=1}^{M} (m_{ki} - \mu_i)(m_{kj} - \mu_j) \tag{4.4}$$

where μ_i is the mean value of variable i. Notice that the correlation coefficients c_{ij} are calculated between values from the prototype vectors instead of original data values to benefit from the noise reduction made by the quantization process.

The weakness of correlation coefficient is that it only works for monotonous dependencies. Ultimately, it would be better to use some dependency measure which allows for non-monotonous relationships. In Publication 5, relative changes in different variables in the local neighborhoods are measured:

$$\frac{\sum_{k \in N_j} |m_{ji} - m_{ki}|}{\sum_{k \in N_j} ||\mathbf{m}_j - \mathbf{m}_k||}.$$

The correlation coefficients between these are found to work well to indicate non-monotonous relationships. Also some other measure could be used, for example mutual information, accuracy of a predictive model, or possibly some measure based on quantization. In [90] vector quantization is utilized to measure the dependencies in order to find groups of independent variables.

After the dependency vectors for the variables have been calculated, conventional clustering and projection algorithms can be used to project, order, or cluster the variables. In Publication 5, SOM, CCA and Sammon's mapping are applied and compared. In Publication 10, the variables are ordered both on a line and on a circle, as well as clustered using agglomerative clustering.

4.2.4 Visualizing data on the map

An important visualization task is to compare individual data samples or data sets with the map. Traditionally this is based on finding the BMU of each data sample from the map. The BMUs of familiar data samples can be used to identify regions from the map. In case of time-series data, the adjacent BMUs can be combined to form a trajectory on the map which can be used in process monitoring to visualize the development of process state over time [70, 128]. For data sets, data histograms are obtained by counting the number of "hits" in each map unit. The distribution of the hits can be used to compare different data sets to each other, as done in the pulp and paper mills case study in Publication 2. Figure 4.11a and b presents the histograms of the distribution of data samples from two different geographical regions on the mills map.

However, the BMU is not the whole truth, since simply indicating it from the map completely ignores the accuracy of the match. Visualizing just the BMU gives wrong impression when:

- The data sample is close to two (or more) different areas of the map, and thus a truer representation would show the response as multimodal.
- The data sample is very far from the whole map, and thus the distances to all map units are almost equal.

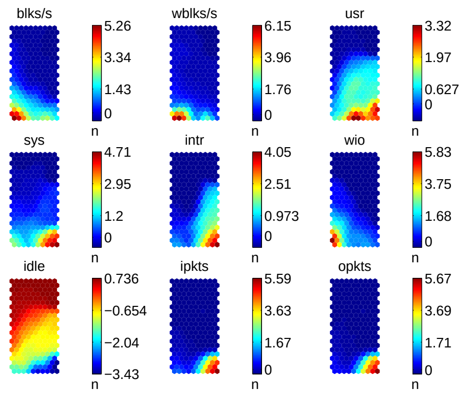


Figure 4.8: Component planes visualizations for system data.

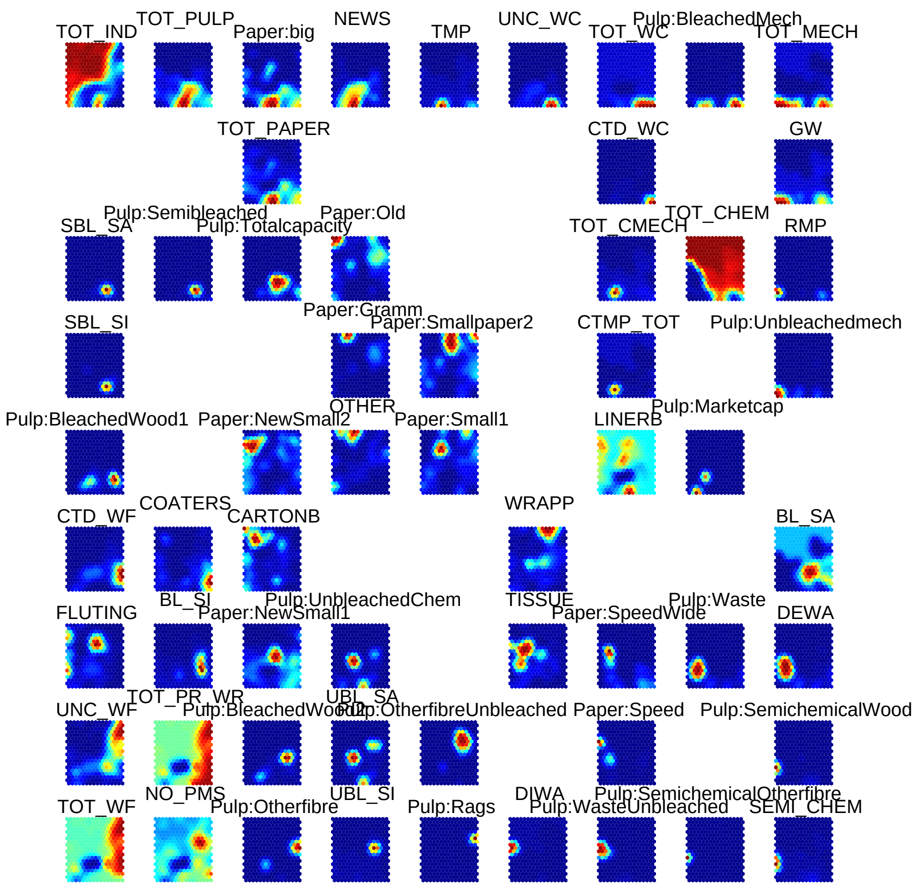


Figure 4.9: Component planes visualizations for mill data. The component planes have been organized such that correlated components are near each other.

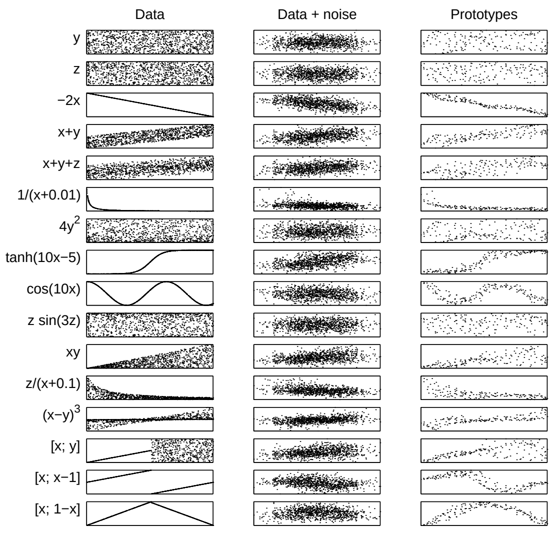


Figure 4.10: Noise reduction performed by the SOM. Scatterplots of the data used in Publication 5: *x* on horizontal axis, the other 16 variables on the vertical axis. On the left, the data with no noise. In the middle, data and Gaussian noise with signal-to-noise ratio 1. On the right, corresponding scatterplots from a map trained with the noisy data.

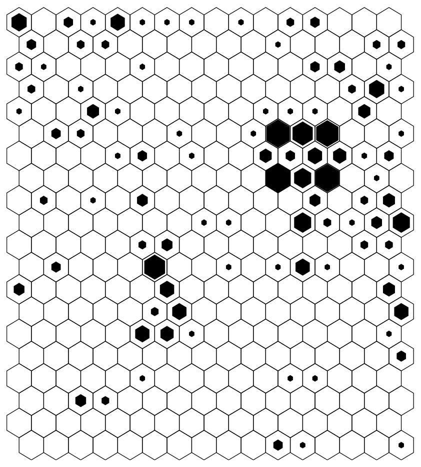
Both cases can occur either when the map does not represent the data very well, or when the data sample is very rare or from a different distribution than the training data. The former case corresponds to verifying the quality of the map, and the latter to novelty or fault detection [27, 70].

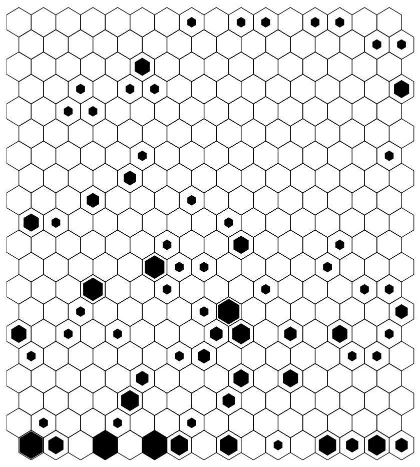
In Publication 3 it is proposed that the response should be calculated and shown such that also the accuracy of the match would be apparent, and two basic ways to do this are proposed. The response of all map units to the data sample(s) can be calculated and visualized. A heuristic response measure that works well is:

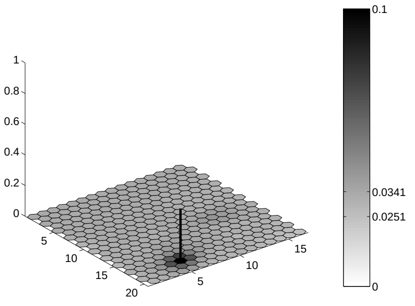
$$r(\mathbf{x}_i, \mathbf{m}_j) = \frac{1}{1 + (||\mathbf{x}_i - \mathbf{m}_j||/a)^2} \tag{4.5}$$

where a is the average quantization error for the training data. An example is shown in Figure 4.11c. In Publication 4, a more principled approach is investigated. A generative mixture model of Gaussian distributions is estimated on top of the SOM, and the response is defined as $r(\mathbf{m}_j, \mathbf{x}_i) = P(\mathbf{m}_j|\mathbf{x}_i)$ (see also Section 4.4). This function is also used in Publication 9 to calculate responses of high-level items on on low-level maps (see Section 5.2).

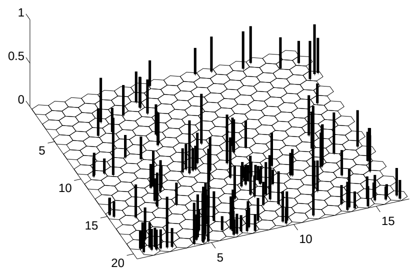
The other way is to position the samples in the visualization so that the accuracy is apparent from either the size or the position of the sample marker. This has the advantage that multiple data samples can be plotted simultaneously, as in Figure 4.11d.


(a) Mills in China


(b) Mills in Scandinavia



(c) One mill



(d) Mills in Scandinavia

Figure 4.11: Visualizations of responses of data on the mills map. In (a and b) data histograms for Chinese and Scandinavian pulp and paper mills: the more there are hits in a map unit, the bigger the black marker. It can be seen that the distributions are remarkably different. In (c) the response of a single mill on the map is shown. The responses of all map units (according to Equation 4.5) are shown as the coloring of the plane. The colorbar on the right shows the minimum, mean and maximum response values (0.025, 0.034 and 0.1, respectively) for the data sample. The vertical line is positioned on the BMU of the data sample and its height equals the ratio between the mean and maximum values of the response (0.034/0.1 = 0.34): values close to zero indicate a good response. In (d) corresponding response indicator bars are shown for all Scandinavian mills.

4.3 Clustering

Clustering algorithms divide, or partition, data into natural groups of objects. The definition of “natural” is a bit vague[4], but usually it means that the objects in a cluster should be internally similar to each other, but differ significantly from the objects in the other clusters.

Most clustering algorithms produce crisp partitionings, where each data sample belongs to exactly one cluster. To reflect the inherently vague nature of clusterings, there are also some algorithms where each data object may belong to several clusters to a varying degree. These are called fuzzy clustering algorithms [7]. Also mixture models [102] can be used to provide such partitionings.

Another way to deal with the complexity of real data sets is to construct a cluster hierarchy. Clustering may depend on the level of detail being observed, and thus a cluster hierarchy may, at least in principle, be better at revealing the inherent structure of the data than a direct partitioning, see Figure 4.12. A hierarchical set of clusters also allows the data to be investigated at several levels of granularity.

In the context of this thesis, cluster analysis is used for two purposes:

- to divide the data into sensible (and crisp) subsets, and
- to gain insight of the structure and contents of the data set through the hierarchy and descriptions of the subsets.

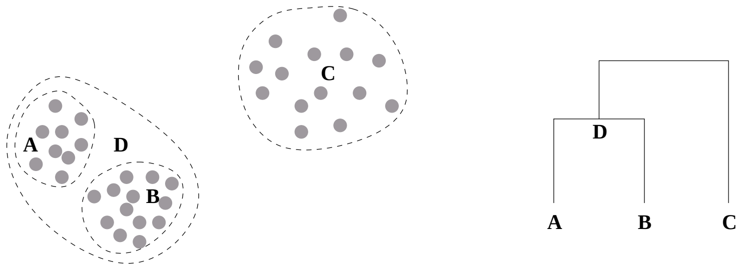


Figure 4.12: Interesting clusters may exist at several levels. In addition to A, B and C, also the cluster D, which is a combination of A and B, is interesting.

4.3.1 Cluster distances

Distance definitions

Clustering algorithms are based on the notion of cluster distance. Two kinds of distances are used: within-cluster distances, and between-clusters distances, see Table 4.1. Within-cluster distances $S(\cdot)$ measure the degree of internal dispersion present in the cluster. Between-clusters distances $d(\cdot,\cdot)$ measure the separation between clusters.

[4]If a human observer partitions a set of objects into groups visually, the results are subject to the Gestalt laws, see page 31. These Gestalt laws are hard to quantify mathematically, and some of them only work well in a 2-dimensional plane (for example closure). Typically, only the proximity property is taken into account in definitions of cluster distances. For this reason partitionings produced by clustering algorithms sometimes appear counter-intuitive.

Table 4.1: Some definitions of within-cluster distances $S(C_k)$ and between-clusters distances $d(C_k, C_l)$; $\mathbf{x}_i, \mathbf{x}_{i'} \in C_k$, $i \neq i'$, $\mathbf{x}_j \in C_l$, $k \neq l$. N_k is the number of samples in cluster C_k and $\mathbf{c}_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$ is the center of cluster C_k.

Within-cluster distance $S(C_k)$	
average	$S_a = \frac{\sum_{i,i'} \|\mathbf{x}_i - \mathbf{x}_{i'}\|}{N_k(N_k-1)}$
nearest neighbor	$S_{nn} = \frac{\sum_i \min_{i'}\{\|\mathbf{x}_i - \mathbf{x}_{i'}\|\}}{N_k}$
centroid	$S_c = \frac{\sum_i \|\mathbf{x}_i - \mathbf{c}_k\|}{N_k}$
variance	$S_v = \sum_i \|\mathbf{x}_i - \mathbf{c}_k\|^2$
Between-clusters distance $d(C_k, C_l)$	
single linkage	$d_s = \min_{i,j}\{\|\mathbf{x}_i - \mathbf{x}_j\|\}$
complete linkage	$d_{co} = \max_{i,j}\{\|\mathbf{x}_i - \mathbf{x}_j\|\}$
average linkage	$d_a = \frac{\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|}{N_k N_l}$
centroid	$d_{ce} = \|\mathbf{c}_k - \mathbf{c}_l\|$
Ward	$d_w = \frac{N_k N_l \|\mathbf{c}_k - \mathbf{c}_l\|}{N_k + N_l}$

The measures can also be divided to two groups according to whether they are local or not. Most measures in Table 4.1 are non-local. The local measures d_s and S_{nn} take into account only the local neighborhood of each point. For example, S_{nn} estimates the density of the cluster by the mean distance between each point and its nearest neighbor. Distances to the neighbors provide information of the local density of the data which can then be utilized in finding cluster borders [30].

The neighborhoods for local distances can be defined as a proximity graph: each node corresponds to one data sample, and is connected to the nodes that form its neighborhood. If the proximity graph is constructed using single linkage, the graph is the minimum spanning tree of the data. Such graph is very sensitive to noise, though. Instead, more fully connected graphs can be used. For example, in the Chameleon clustering algorithm [62] k nearest neighbors are used to construct the proximity graph. An overview of methods for constructing proximity graphs is given in [30].

Proximity graphs can be utilized in measuring the cluster distances. For example, an extended version of d_s measures the average distance between nodes in the clusters:

$$d_{cl}(C_k, C_l) = \frac{\sum_i \sum_j \mathbf{1}(j \in \mathcal{N}_i) \|\mathbf{x}_i - \mathbf{x}_j\|}{\sum_i \sum_j \mathbf{1}(j \in \mathcal{N}_i)}, \tag{4.6}$$

where $\mathbf{1}(j \in \mathcal{N}_i) = 1$ if there is a connection between data samples i and j in the proximity graph, and 0 otherwise. Alternatively, clusters can be constructed directly from proximity graphs by cutting edges that are determined to be on the borders between clusters. Then, each separate sub-graph forms one cluster.

Normalization

Most distance measures between clusters are based on distances between individual data vectors, or their representatives (in effect, cluster centers), and thus they are sensitive to the scales of the variables. It is easy to come up with examples where the clustering result can be considerably changed by a simple linear rescaling of the variables (see for example [71] page 5). Therefore, apart from the case when the original scales of the variables are directly comparable with each other, some kind of rescaling or standardization procedures are normally recommended prior to the clustering.

The most common standardization procedure is to treat all variables independently and transform each to Z-scores by subtracting the mean and dividing by the standard deviation of each variable (see Section 3.1.3). Some studies have found that scaling by range:

$$x_{ij} = \frac{x_{ij} - \min_j(x_{ij})}{\max_j(x_{ij}) - \min_j(x_{ij})}$$

works better than Z-scoring in clustering [106]. Scaling by range is, however, much more sensitive to outliers than Z-scoring. In view of the results in Publication 8, Z-scoring also provides a more equal starting point for each variable. Z-scoring removes the effect of scale, and thus only variable characteristics and dependencies between variables effect the quantization result.

Cluster validity

Since clustering results are usually open to at least some interpretation, it is important to accompany clusters with validity information. Validity measures are also important because most clustering algorithms do not produce a single partitioning of the data, but offer several with different numbers of clusters. Hierarchical clustering algorithms work inherently in this manner by finding the clusters iteratively, adding or removing one cluster at a time. Most non-hierarchical algorithms require the number of clusters as a predefined parameter. If this is not known beforehand, several values need to be experimented with. When faced with different options, one has to apply validity measures to find out which of them is the best[5]. In [105] 30 validity indexes are presented and evaluated using artificial data sets.

A widely adopted definition of a good cluster(ing) is one that has small within-cluster distances $S(C_i)$ and large between-clusters distances $d(C_i, C_j)$. One way to use this definition is to minimize (or maximize the inverse of):

$$\frac{S(C_i) + S(C_j)}{d(C_i, C_j)}, \ \forall j \neq i \tag{4.7}$$

However, this leaves much room for variation. Figure 4.13 illustrates two ways to define the within- and between-clusters distances in such a measure. The one on the left represents the widely used Davies-Bouldin index according to which the best clustering minimizes:

$$I_{DB} = \frac{1}{C} \sum_{i=1}^{C} \max_{j \neq i} \left\{ \frac{S_c(C_i) + S_c(C_j)}{d_{ce}(C_i, C_j)} \right\}, \tag{4.8}$$

where C is the number of clusters. The index gets low values for clusters which are compact and far from the other clusters. However, since the within-cluster distances are measured using S_c, the Davies-Bouldin index makes the implicit assumption that the clusters are hyper-spheres.

The cluster validity measure illustrated on the right in Figure 4.13 is used in Publication 7. It is based on local distance measures S_{nn} and d_s and compares the smallest gap between each pair of clusters to their internal densities. When the gap between

[5]Of course, cluster validity measures can also be used as part of an actual clustering algorithm as in [62].

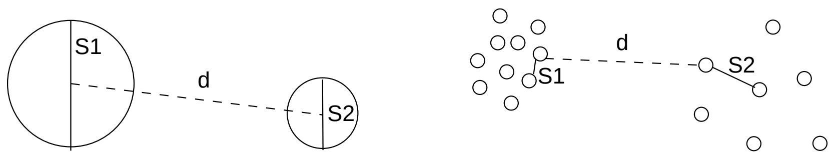


Figure 4.13: Different criteria for cluster validity. The left one measures separation between cluster centers d_{ce} with respect to their internal size S_c, while the one on the right measures the gap between the clusters d_s with respect to their internal density S_{nn}. For definitions of the distance measures, see Table 4.1.

clusters is bigger than the average gap between samples in each cluster, the cluster is considered valid. The distance measures S_{nn} and d_s are very sensitive to noise, though. In Publication 7 they are modified slightly to produce more robust estimates.

4.3.2 Clustering algorithms

Hierarchical vs. partitional

There are two fundamentally different approaches to clustering: hierarchical and partitional [57]. Hierarchical clustering algorithms find clusters one by one. The hierarchical methods can be further divided to agglomerative and divisive algorithms, corresponding to bottom-up and top-down strategies. Agglomerative clustering algorithms merge clusters together one at a time to form a clustering tree which finally consists of a single cluster, the whole data set. The algorithms consist of the following steps:

1. initialize: assign each vector to its own cluster, or use some initial partitioning provided by some other clustering algorithm
2. compute distances $d(C_i, C_j)$ between all clusters
3. merge the two clusters that are closest to each other
4. return to step 2 until there is only one cluster left

Divisive algorithms are similar but work in the opposite direction starting from a single cluster and dividing each cluster to sub-clusters until all vectors are in a cluster of their own.

Partitional clustering algorithms, on the other hand, divide a data set directly into a (given) number of clusters, typically by trying to minimize some criterion or energy function. The number of clusters is usually predefined, but it can also be part of an energy function [14]. Partitional methods are less sensitive to imperfections — noise and outliers — in the data than hierarchical methods [99]. On the other hand, many partitional methods — especially those based on some kind of global energy function — make implicit assumptions of the form of the clusters.

Computational efficiency

Most clustering algorithms are computationally very intensive: the computational complexity is often at least $O(N^2d)$. Many of recent advances in clustering attempt to reduce the computational cost to make clustering of very large data sets — with millions of samples — possible.

In the BIRCH clustering algorithm [150], the computational complexity is reduced by constructing a hierarchical organization of the clusters. This hierarchy is utilized to assign new samples to the clusters so that the data sample only needs to be compared to a small fraction of the clusters instead of all of them. This resembles the winner search procedures used in TS-SOM [86] variant of the SOM. In addition, for each cluster a "clustering feature" vector

$$\left[N_i \, , \sum_{\mathbf{x} \in C_i} \mathbf{x} \, , \sum_{\mathbf{x} \in C_i} \mathbf{x}^2 \right]$$

is maintained which reduces the computational load of distance calculations.

In CURE algorithm [40], the computational complexity is reduced by making the clustering in two phases. First, the data set is divided to p subsets, and each subset is pre-clustered to c clusters. In the second phase, the cp clusters in the subsets are clustered. Computational load is less because the cost of clustering N samples is higher than the cost of clustering p data sets of size N/p plus clustering the cp intermediate clusters.

In addition, CURE algorithm reaches significant reducements in computation time with random sampling. In the experiments based on artificial 2-dimensional data sets, a sample size of 2.5% was sufficient to consistently produce correct clustering results. Random sampling is of course applicable in conjunction with any clustering algorithm. In [13] it was pointed out that random sampling could also be used to construct good initializations for the clustering algorithms, making them converge faster and to better solutions.

4.3.3 Clustering based on SOM

Clustering is one of the main applications for SOM in data mining. In Publication 7, different ways to cluster data using SOM are discussed. The validity of using SOM in clustering is investigated by comparing SOM-based clustering results to those produced by clustering the data directly. The SOM-based clustering approach is considerably faster than clustering the data directly, and the correspondence between the clustering results is good.

Clustering using the SOM

Like CURE, clustering using SOM is a 2-phase strategy:

1. A SOM is trained. The Voronoi regions of the map units provide an intermediate partitioning of the data.
2. The map units of the SOM are clustered. Each data sample belongs to the same cluster as its BMU.

In order to find the natural clusters of the data, the number of map units in the trained SOM should be much bigger than the expected number of clusters $M \gg C$ [132]. This is because the neighborhood function draws neighboring map units together, and thus neighboring map units reflect the properties of the same rather than different clusters[6].

Most algorithms proposed for clustering the SOM simply regard the SOM as a set of vectors, and apply standard clustering algorithms, for example k-means or some

[6]Unless the neighborhood radius is decreased to zero $\sigma \to 0$ during training. However, in this case the SOM equals k-means algorithm.

agglomerative clustering algorithm to cluster the prototype vectors [140]. However, there are also algorithms which take SOM neighborhoods into account. Murtagh has proposed the usage of a neighborhood constraint to make sure that the clusters form continuous areas on the map [108]. An agglomerative clustering algorithm is used to cluster the map units, but only those candidates are considered for merging for which the conjunction of neighborhood sets is not empty:

$$(\bigcup_{i \in C_k} \mathcal{N}_i) \bigcap (\bigcup_{j \in C_l} \mathcal{N}_j) \neq \emptyset \tag{4.9}$$

This constraint may be a problem if the map is folded such that some areas of the data space are separate on the map although they are connected in the original space. Alternatively, a cluster distance function can be used which takes the neighborhoods into account, for example:

$$d_{ne}(C_k, C_l) = \frac{\sum_{i \in C_k} \sum_{j \in C_l} h_{ij} \|\mathbf{m}_i - \mathbf{m}_j\|^2}{\sum_{i \in C_k} \sum_{j \in C_l} h_{ij}}. \tag{4.10}$$

Like d_{cl} in Eq. 4.6, d_{ne} is a local measure of cluster distance since h_{ij} only gets large values for neighboring map units.

Distance matrix -based clustering

Distance matrices are often used to visualize the cluster structure of the SOM. Because distance matrices show the local density of the data, this is basically a mode-seeking approach. High values of the distance matrix indicate cluster borders, whereas low “valleys” indicate clusters [134]. These are utilized to select areas from the map by hand. Recently, Vellido *et al.* proposed a simple algorithm to do distance matrix based clustering automatically [135]. The local minima of the distance matrix can used to identify cluster centers from the SOM. In Vellido’s approach, the rest of the map units were then assigned to the cluster whose center was closest. This algorithm is simple and fast, but it also makes the implicit assumption that the border between two clusters lies on the middle-point between their cluster centers. In Publication 10, an enhanced version based on region-growing is proposed:

1. Local minima of the distance matrix are found. This is done by finding the set of map units $\{i\}$ for which:

   $$f(\mathcal{D}_i) \leq f(\mathcal{D}_j) \; \forall j \in \mathcal{N}_i,$$

   where $f(\mathcal{D}_i)$ is some function of the set of distance values associated with map unit i, for example mean or median.

   The set of local minima may have units which are neighbors of each other. Only one from each such group should be retained.

2. Initialization. Let each local minimum be one cluster: $C_i = \{\mathbf{m}_i\}$. All other map units $\mathbf{m}_j$ are left unassigned.

3. Calculate distance $d(C_i, \{\mathbf{m}_j\})$ from each cluster C_i to (the cluster formed by) each unassigned map unit $\mathbf{m}_j$.

4. Find the unassigned map unit with smallest distance and assign it to the corresponding cluster.

   Two optional constraints can be used to limit the growth of the clusters:

- The neighborhood constraint: only those unassigned map units are considered for merging which are neighbors of the units in the clusters. This ensures that the clusters form continuous areas on the map.
- Cluster border constraint: map units on borders between clusters may have been identified beforehand using, for example, presence of empty map units, as proposed in [151]. Connections to such border units can be removed, thus creating barriers for the region-growing procedure.

5. Repeat from step 3 until no more connections can be made.
6. If there are any cluster border units, assign them to the same cluster as the closest (neighboring) map unit prototype.

This procedure provides a partitioning of the map into a set of base clusters, the number of which is equal to the number of local minima on the distance matrix. A problem is that there may be some minima which are the result of random variations in the data rather than real local maxima of the probability density function. In such a case, however, the local minima will be quite shallow, and they can be pruned out. The best way to actually do this is an ongoing research issue.

4.3.4 Cluster hierarchy

Irrespective of how the base partitioning of the data is done, agglomerative clustering algorithms can be used to construct the cluster hierarchy starting from those base clusters.

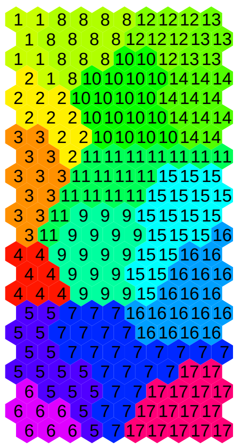
However, most agglomerative algorithms produce binary trees which may not be representative of the true structure. If in reality, a super-cluster consists of three (or more) sub-clusters, the binary tree will have one (or more) extra intermediate clusters which should be pruned out. This can be done by hand using some kind of interactive tool [11], or in an automated fashion for example as in Publication 10:

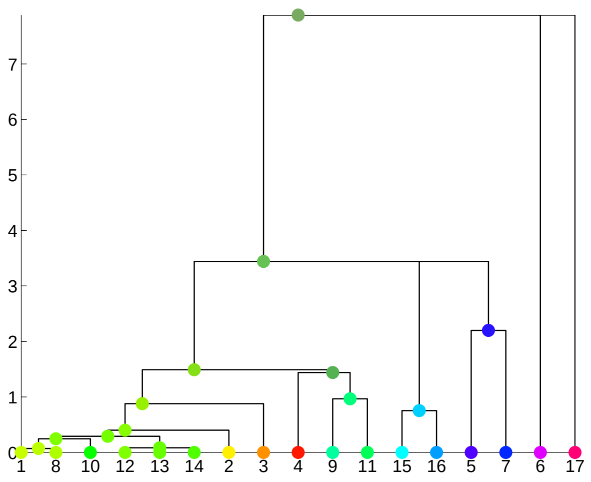
1. Start from root cluster.
2. For the cluster under investigation, generate different kinds of sub-cluster sets by replacing each cluster in the sub-cluster set by its own sub-clusters.
3. Each sub-cluster set defines a partitioning of the data in the investigated cluster. Investigate each partitioning using Davies-Bouldin index I_{DB} (Eq. 4.8).
4. Select the sub-cluster set with minimum value for I_{DB}, and prune the corresponding intermediate clusters.
5. Select an uninvestigated and unpruned cluster, and continue from step 2.

Figure 4.14 shows the final clusters and the cluster hierarchy for the example data sets.

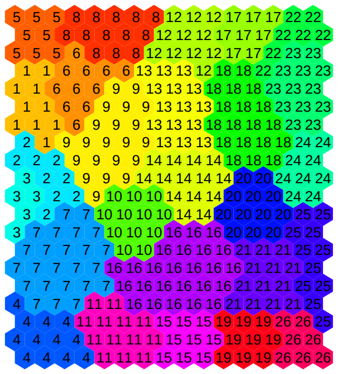
4.3.5 Cluster characterization

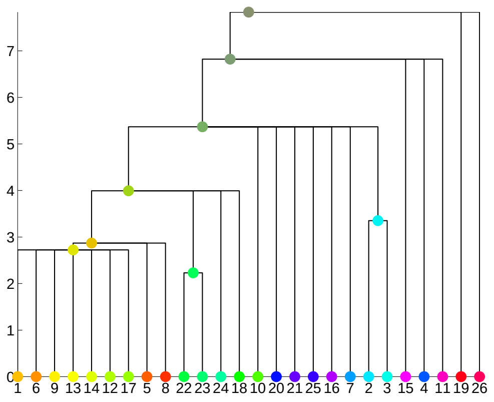
In data exploration it is not sufficient to know the number and hierarchy of clusters. The question that immediately follows is, what are the clusters like? This is investigated in Publications 9 and 10.


(a) System: base clusters



(b) System: hierarchy


(c) Mills: base clusters



(d) Mills: hierarchy

Figure 4.14: Clustering results for the system (a, b) and mills (c, d) maps. Figures (a) and (c) show the base clusters found by the distance matrix based algorithm, and figures (b) and (d) the pruned cluster hierarchy built on top of the base clusters. The clusters are colored with the corresponding cluster colors. The colors of the super-clusters are calculated as averages of their sub-clusters.

Significant variables

Cluster descriptions must both tie the cluster in with the rest of the data, and tell about the internal properties of the cluster. Descriptive statistics can be used to list the values typical for each cluster, and associated indexes — for example ratio between mean values in the cluster and in the whole data — to relate them to the rest of the data. However, considering that the number of clusters and the number of variables may be quite large, this may result in an impractically lengthy listing: accurate, but not very useful from data exploration point of view. Therefore, a good approach is to derive some measure of significance and use it to rank the variables [67, 91, 112, 133].

An indication of significance is deviation from the excepted. The variables can be ordered by measuring the difference between distributions of the values in the cluster versus the whole data. However, since the number of data samples in the cluster may be quite low, such estimates are rather noisy. Instead, some heuristic criterion can be used, for example standard deviation:

$$s^c_{ik} = \frac{\sigma_{ik}}{\sigma_k}, \tag{4.11}$$

where σ_{ik} is the standard deviation of variable k in cluster i, and σ_k is the standard deviation of variable k in the whole data. This measures the relative importance of the variables within each cluster, or how well the variable k characterizes the values in the cluster i.

Even if a variable is characteristic for a cluster it may not differentiate it well from the other clusters. A second measure is:

$$s^d_{ik} = \frac{(C-1)s^c_{ik}}{\sum_{j=1,j\neq i}^{C} s^c_{jk}}, \tag{4.12}$$

where C is the number of clusters. This measures the discriminating property of the variable: it gives high values when a variable is characteristic for only one or a few clusters.

Characterizing and differentiating rules

Another frequently employed method is to form characterizing rules [1, 44, 131, 133] to describe the values in each cluster. The rules are usually based on intervals of variable values:

$$R_i : \mathbf{x} \in C_i \Leftrightarrow x_k \in [\alpha_k, \beta_k] \tag{4.13}$$

where x_k is the value of variable k in sample vector $\mathbf{x}$, C_i is the investigated cluster, and $[\alpha_k, \beta_k]$ is the range of values allowed for the variable according to the rule. These rules may concern single variables like R_i above, or be conjunctions of several such rules in which case the rule forms a hyper-cube in the input space[7].

The main advantage of conjunctive rules is that they are compact, simple and therefore easy to understand. The problem is of course that clusters often do not coincide well with the rules since the edges between clusters are not necessarily in parallel with the edges of the hyper-cube[8]. In addition, the cluster may include some uncharacteristic

[7] In the NDM tools reported in [132], the rules can also be combined with “or” or a majority-vote.

[8] In conceptual clustering [104], the distances between clusters are defined as distances between their descriptions. Thus, they could be used to construct clusters which are more in line with such hyper-cubes. However, conceptual clustering techniques are usually computationally rather heavy. In addition, it is not clear whether such distance measures produce natural clusters.

points or outliers. Therefore the rules should be accompanied by validity information. The rules R_i can be divided to two different cases, characterizing and differentiating rules:

$$\begin{aligned} R_i^c: &\quad \mathbf{x} \in C_i \Rightarrow x_k \in [\alpha_k, \beta_k] \\ R_i^d: &\quad x_k \in [\alpha_k, \beta_k] \Rightarrow \mathbf{x} \in C_i. \end{aligned}$$

The validity with respect to each case can be measured using confidence: $P_i^c = P(x_k \in [\alpha_k, \beta_k] \,|\, C_i)$ and $P_i^d = P(C_i \,|\, x_k \in [\alpha_k, \beta_k])$, respectively.

To form the characterizing rules — in effect to select the low and high limits of the range — one can use statistics of the values in the clusters as in [133]. Another approach is to optimize the rules with respect to their significance. The optimization can be interpreted as a two-class classification problem between the cluster and the rest of the data (see Figure 4.15). The boundaries in the characterizing rule can be set by maximizing a function which gets its highest value when there are no miss-classifications, for example:

$$s_1 = \frac{a+d}{a+b+c+d}, \tag{4.14}$$

$$s_2 = \frac{a}{a+b}\frac{a}{a+c}, \tag{4.15}$$

$$s_3 = \frac{a}{a+b+c}, \tag{4.16}$$

where a, b, c and d are from the truth table in Figure 4.15. The first function s_1 is simply the classification accuracy. It has the disadvantage that if the number of samples in the cluster is much lower than in the whole data set (which is very often the case), s_1 is dominated by the need to classify most of the samples as false. Thus, the allowed range of values in the rule may vanish entirely. As pointed out in [3], traditional rule-based classification techniques are not well suited for the characterization task.

When characterizing the (positive) relationship between rule R and cluster C, the samples belonging to d are not really interesting. The two latter measures consider only cases a, b and c. The second measure s_2 is the product of the confidences $s_2 = P_i^c P_i^d$. The third measure s_3 is the classification accuracy when the case d is ignored. Notice that $s_3 \approx s_2$ when $a \gg b + c$.

Apart from characterizing the internal properties of the clusters, it is important to understand how they differ from the other, especially neighboring clusters. For the neighboring clusters, the constructed rules may be quite similar, but it is still important to know what makes the clusters different. To do this, rules can be generated using the same procedure as above, but taking only data samples in the two clusters into account. In this case, however, both clusters are interesting, and therefore s_1 should be used.

In Figure 4.16 and Table 4.2 an example of the rules for one cluster in the system data is shown. In this case, the conjunctive rule formed of four variables is very good: the rule holds for the cluster only ($P^d = 100\%$) and it also characterizes the cluster very well ($P^c = 96\%$).

4.4 Local modeling using SOM

In this section, local modeling using SOM is shortly discussed. The presentation is very short, and only attempts to bring forth the most important issues to consider when using SOM as a basis for modeling.

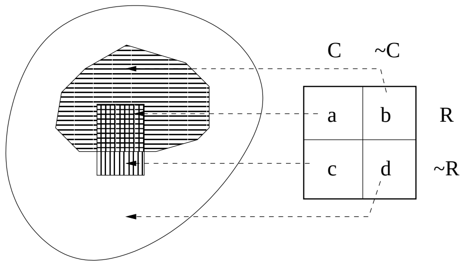


Figure 4.15: The four-way truth table of cluster *C* and rule *R*. The cluster is the horizontally shaded area, and the rule (or classification model) is the vertically shaded area. On the right is the corresponding confusion matrix: *a* is the number of samples which are in the cluster, and for which the rule is true. In contrast, *d* is number of samples which are out of the cluster, and for which the rule is false. Ideally, the off-diagonal elements in the matrix should be zero.

The SOM — or more generally: any clustering or vector quantization method — partitions the data space into small segments (map units or clusters) which can be easily utilized in building local models, as done in Publications 1 and 4. A scheme for constructing and using the local models used in Publication 1 is depicted in Figure 4.17. It is based on four phases:

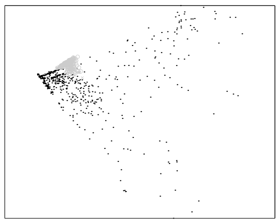
1. Partitioning the input space.
2. Building local data sets.
3. Calculating local models.
4. Predicting new values by selecting and applying the local model(s).

In Publication 4, three ways to build to the local data sets (phase 2) are compared in a probability density estimation problem. Below the four phases are discussed in more detail.

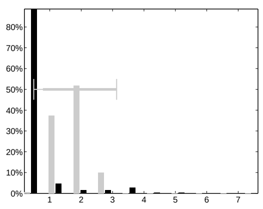
Input space partitioning

First the data space is partitioned using, for example, vector quantization. Normally quantization is based on unsupervised learning, and thus the segments are concentrated where there is a lot of data and not, for example, on the areas where more detailing power would be needed for the modeling task. Thus, although the modeling results can be quite good, they are in general suboptimal. There are some related methods and techniques which use (partial) supervision to position the prototypes better:

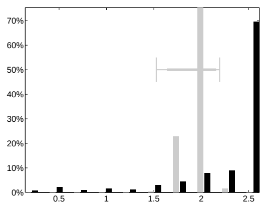
- In supervised SOM [61], the output classes are used together with the input variables to form a quantization which conforms better to the output class distribution.
- In constrained topological mapping (CTM) [17], the scales of individual variables are dynamically adjusted to reach good learning results.



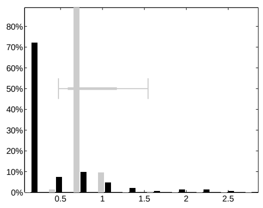
(a) System: cluster 24



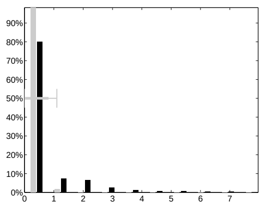
(b) Rule for `intr`



(c) Rule for `idle`



(d) Rule for `usr`



(e) Rule for `blck/s`

Figure 4.16: Cluster 24 of the system data (combination of base clusters 15 and 16). In (a) the PCA-projection of the data is shown, with cluster 24 indicated by the grey markers and the rest of the data with black markers. In (b-e) are the histograms corresponding to the most significant rule (see Table 4.2). The grey vertical bars are the histogram for cluster 24, and the black the histogram for the rest of the data. The horizontal bar indicates the range allowed by the rule (thick part) and the real range of values in the cluster (thin part).

Table 4.2: Rule summary for cluster 24 in the system data. Variables are listed in order of decreasing significance as measured by s_2. The columns in the middle indicate the properties for the variable-wise rules, and the columns on the right for a conjunctive rule formed of the indicated variables starting from the top. The "diff" columns are confidences in the differentiating property of the rule P^d and "char" column in the characterizing property P^c.

Variable	**Rule**	**diff**	**char**	s_2	**diff**	**char**	s_2
			single			**cumulative**	
`intr`	[0.78,3.1]	75%	99%	0.745	75%	99%	0.745
`idle`	[3.3,4.3]	65%	98%	0.639	87%	98%	0.856
`usr`	[1.2,2.3]	61%	98%	0.605	90%	98%	0.885
`sys`	[0.71,1.4]	69%	62%	0.425			
`blks/s`	<0.82	22%	98%	0.217	100%	96%	0.96
`wio`	<1.1	21%	100%	0.214			
`ipkts`	<0.97	20%	100%	0.201			
`opkts`	<0.97	20%	100%	0.2			
`wblks/s`	<1.4	20%	100%	0.198			

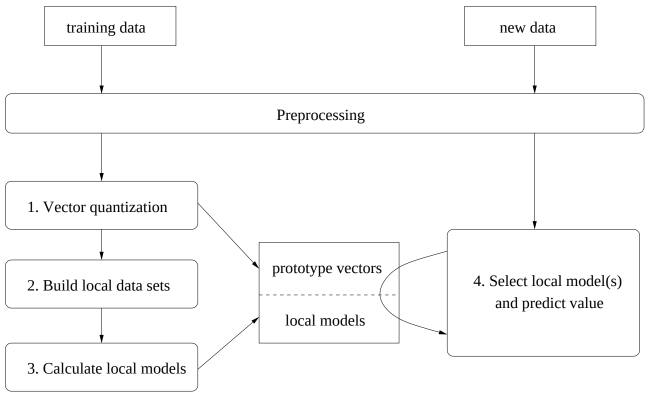


Figure 4.17: A scheme for constructing and using local models. As opposed to most other SOM-based modeling approaches, in this approach the training of local models is decoupled from the map training. This is an advantage in data exploration, since new models can be trained without retraining the SOM.

- Recently, Sinkkonen and Kaski have proposed a method which utilizes the probability density of the output variable to guide the quantization process in order to make the map quantization and organization better reflect the changes in the output variable [125].
- The Learning Vector Quantization (LVQ) [80] is a supervised classification algorithm which, like SOM, is based on prototype vectors. Unlike SOM, however, LVQ uses class information to refine the positions of the prototypes on the borders between classes.

Local data sets

Local data sets L_j are constructed for each prototype vector. Primarily, the data in the Voronoi set of the particular prototype vector is used $L_j = V_j$ but often this set is too small — even empty. In order to make reliable local models, the local data set is augmented with data from nearby prototypes:

$$L_j = \bigcup_{k \in \hat{\mathcal{N}}_j} V_k, \tag{4.17}$$

where $\hat{\mathcal{N}}_j$ is some neighborhood set of map unit j. Notice that the normal neighborhood definition of SOM, which is defined in the output space, may not be ideal for defining neighborhoods for local modeling. In the comparison study in Publication 4, it is clearly inferior to using the set of six map units with closest prototypes — in effect, defining the neighborhood in the input space. Also in Publication 1 the local data sets are augmented from the Voronoi sets of closest prototypes instead of neighbors on the grid.

Alternatively, the whole data set can be used but with a set of weighting factors w_i determined locally for each data sample, as done in Publications 4 and 9. The local data set is then:

$$L_j = \{(\mathbf{x}_i, w_i)\} \ \forall i. \tag{4.18}$$

In Publication 4, the neighborhood function values between the prototype vector j and the BMU b_i of the data sample i are used as weights $w_i = h_{b_i j}$. In Publication 9, a probability density is estimated on top of the SOM, and the probabilities $w_i = p(\mathbf{x_i}|\mathbf{m}_j)$ are used to weight the class probabilities in the frequency components.

Local models

Local models $y = f_j(\mathbf{x})$ are calculated using the local data sets L_j. Basically, models can be divided to three different types: regression, classification and probability density estimation [18].

- In regression, the simplest local model is simply a reference value calculated as a (weighted) average of the output values associated with each input vector $\mathbf{x}_i$. Notice that when $w_i = h_{b_i j}$, this corresponds to the update rule of the Batch Map algorithm (Eq. 3.4).

  Another frequently used option is to use linear models. Such models have been constructed, for example, in [25, 110, 115, 146] as well as in Publication 1. More complex models are not generally used due to the danger of overfitting to the relatively small amount of data in each local data set.

- In classification problems, an output class can be assigned to each map unit based on a voting among the data samples in the local data set. Alternatively, the relative probabilities of each class can be calculated.

- A SOM-based probability density model proposed by Lampinen and Kostiainen was described in Section 3.1.3 [93]. The probability density estimation in Publication 4 follows an earlier model investigated by Hämäläinen [43]. The density of the data is estimated as a mixture of Gaussians with centers at the map prototypes and with diagonal covariance matrices. In Publication 4, the *a priori* probabilities are estimated using the sizes of the Voronoi sets N_j of each map unit weighted by the neighborhood function. The variances of each variable are calculated using a weighted average of the squared difference between the value in the prototype and each sample.

Predicting new values

In order to predict the output value y associated with given input data vector $\mathbf{x}$, the local model must first be selected. This is usually done by selecting the local model associated with the best-matching prototype vector b_i:

$$y = f_{b_i}(\mathbf{x}). \tag{4.19}$$

Alternatively, a weighted average of the outputs of multiple or even all local models can be used. First, a response $r(\mathbf{m}_j, \mathbf{x}_i)$ of each prototype to the data sample is calculated, and these responses are used to weight the outputs y_j of each prototype:

$$y = \frac{\sum_{j=1}^{M} y_j r(\mathbf{m}_j, \mathbf{x}_i)}{\sum_{j=1}^{M} r(\mathbf{m}_j, \mathbf{x})}. \tag{4.20}$$

This kind of model can be interpreted as a mixture of experts.

Local modeling using SOM is in certain aspects very close to radial basis function (RBF) networks (see for example [8]). In these, also, the data space is first quantized into a set of smaller regions. Typically, this is done in unsupervised manner using some vector quantization algorithm, for example *k*-means. After this, basis functions $K(||\cdot||)$ — for example Gaussian kernels — are associated with each of the prototypes $\mathbf{m}_j$. Selecting the basis functions and their parameters, for example the variance σ of a Gaussian kernel, corresponds to phase 2 discussed above. The output of an RBF is a linear combination of the basis function values:

$$y = \sum_{j=1}^{M} f_j K(||\mathbf{m}_j - \mathbf{x}||) + f_0. \tag{4.21}$$

Therefore, once basis functions have been selected, the coefficients f_j can be found easily using linear algebra. Models build on top of a SOM can naturally also be constructed in the same way.

4.5 Discussion

The SOM is primarily a tool for unsupervised analysis of data. It combines two properties — quantization and projection — which support different data analysis tasks. The projection property is essential for visualization, while quantization forms the basis for SOM-based clustering and modeling. Below, the strengths and weaknesses of the SOM-based methods proposed and investigated in the publications are discussed.

Visualization

As a visualization method, the SOM provides a projection upon which different kinds of properties of the data can be shown. In Publication 3, an overview and categorization of SOM-based visualization methods is given, and a few enhancements are proposed.

- The component plane reorganization (see Section 4.2.3) is used to find pairs and groups of related variables. The technique is very useful when dealing with a large number of variables. Furthermore, the underlying principle — to calculate a set of features which indicate the dependencies between variables, and then apply projection or clustering techniques to these variable-features — can be easily extended as done in Publication 10.

  Publication 5 investigates different options to do the reorganization and compares the SOM to PCA and Sammon's mapping in the reorganization task. The projection methods do not have big differences in their performance. Much more important is selection of the dependency measure between variables. The publication investigates two different measures: one works well for monotonous dependencies but very poorly for non-monotonous dependencies. The other works relatively well in both cases, but only for variables which depend only on a single primary variable. To be really useful, a measure working in all cases would be needed.

  A shortcoming with Publication 5 is that the SOM-based approach is not compared to other visualization techniques used for correlation hunting, for example correlation matrix, scatter plot matrices, or parallel axis plots. It is not clear whether the SOM is better or worse than other approaches. However, the reorganization of variables — in one way or another — is useful in conjunction with most such visualization techniques.

- Publication 3 proposes several new ways to visualize responses of data samples on the SOM such that the localization quality is also shown. This is important because in some cases the responses may be multimodal, or otherwise poorly localized. Except for the pdf-estimation based technique investigated in Publication 4, the techniques are heuristic. A weakness of the publications is that the performance of the methods in real data mining cases is not demonstrated.

A problem with visualization methods in general is their validation. Their usability is highly dependent on issues which have nothing to do with the method itself, such as prior knowledge and expertise of the user, and sophistication of the visualization software. Without extensive usability tests, the methods can only be motivated by heuristic arguments and their evaluation occurs through practical experience. Therefore, it is important to offer easily usable tools for using the techniques. The SOM Toolbox introduced in Publication 6 offers such tools for the Matlab computing environment.

Clustering

Clustering is one of the main applications of the SOM. Apart from visualization based clustering methods, in clustering the projection property of the SOM is a weakness because it interferes with the prototype formation. Therefore, if clustering is the primary aim, it might be prudent to use some variant of SOM where the rigid output grid has been discarded.

Clustering using the SOM is investigated in Publications 7 and 10. Besides clustering algorithms, these publications deal with cluster hierarchy, and cluster characterization.

- As a clustering method, the SOM resembles many recently developed algorithms in that it is a two-phase approach: the data is first preclustered, and then the preclusters are clustered. The results of the two-phase clustering procedure may be slightly different from clustering the data directly because the preclustering procedure is different from the actual clustering algorithm. Publication 7 compares SOM-based methods with clustering the data directly. The comparison indicates that the knowing the SOM-based clustering gives much information of what the direct clustering result would have been. However, the results are not conclusive. The primary motivation for using the SOM in clustering is the reducement in computational load. However, recently several other algorithms have been proposed which are also computationally feasible with very large amount of data. In future work, it would be important to compare the SOM-based clustering approaches to these methods.

  The clustering algorithms used in Publication 7 are based on traditional approaches: agglomerative and *k*-means clustering. Publication 10, on the other hand, presents a novel family of SOM-based clustering algorithms. The family is an extension of the work by Vellido [135]. Because of the scope of the publication, a comparison to other algorithms is not presented, so it is not shown whether the presented algorithms give any advantages over earlier approaches.

- In both publications 7 and 10, forming of cluster trees is emphasized because of complexity of real world data and the usefulness of multiple investigation levels in data exploration. A weakness of the publications is that a definition for "optimal" cluster tree is not given. Instead, the publications simply present some heuristical methods and measures for finding interesting clusters from cluster trees.

- Crucially important from data exploration point of view is characterization of the found clusters. In Publication 9, some measures for finding significant variables for each cluster are presented. These are closely related to the keyword selection measures proposed by Lagus and Kaski [91], the difference being that they are slightly more general.

  In order to generate more detailed information, rules can be constructed. In [132], an algorithm called sig* is used to find significant variables for each cluster, and then to construct characterizing and differentiating rules for the clusters. A similar approach is used in Publications 9 and 10. To be able to characterize small clusters as well as large clusters, the publications present some novel rule significance measures. A weakness of the proposed approach is that the algorithm used to find the rules is heuristic, and it does not even try to find a globally optimal rule for each cluster. The reason for this is that in the implemented report generation system, the computational load needs to kept low and therefore finding a "good" rule is sufficient.

Modeling

Publications 1 and 4 deal with local modeling using the SOM. Both projection and quantization performed by the basic SOM are performed in an unsupervised manner, and thus SOM ultimately forms a suboptimal basis for modeling. While the modeling results in the publications are relatively good, they are inferior to best results achieved by the methods used for comparison.

- In Publication 1 the SOM combined with local linear models is used for time-series prediction. The model architecture resembles closely that of local linear mappings (LLM) [115], the difference being that the models are constructed afterward, rather than simultaneously with the SOM, and in how the local data sets for the models are generated.

  The prediction accuracy of the SOM is equal or better to the compared approaches. However, the reported cases are not all directly comparable. For example, direct cascade architecture combined with local linear mappings (DCA+LLM) is basically very similar to the investigated approach. Its error is slightly higher than the SOM, with only half as much training samples. It can be assumed that with equal number of training samples the DCA+LLM method would be equal or better than the SOM. In addition to training set sizes, also the varying number of model parameters complicates the comparison a bit.

  From modeling point of view, a weakness of the method used in Publication 1 is that the output variable is included in the quantization. Since information of the output variable cannot be used in the prediction phase, this inclusion increases the complexity of the model. However, this does not matter much, since the scaling factor is so small as to effectively remove its effect in quantization.

- The probability density estimation in Publication 4 is based on the reduced kernel density estimators (RKDE) proposed by Hämäläinen [43], with slight differences in how the local kernel parameters are estimated, and how the local data sets are augmented from the neighbors of the map unit. The results of this SOM-based approach are slightly worse than a gaussian mixture model trained with expectation maximation.

In [43], it is noted that correct network topology is often important for RKDE-models. This is confirmed by the results in Publication 4: much better results can be achieved by redefining the neighborhoods in the input space, rather than using the original neighborhoods on the map grid.

Chapter 5

SOM-based data exploration

In this chapter, the SOM-based data exploration process is described in more detail to indicate what the methods introduced in the previous chapter are used for. Exploration of table-format data is divided into a number of sub-tasks, each with its own set of analysis methods. The need and requirements for automated application of these data exploration methods is discussed, and a system implementing this kind of automated exploration is shortly described. In addition to analysis of a single table of data, also analysis of a set of hierarchically organized data tables is considered, and it is shown how the same analysis methodology can be applied in both cases.

5.1 Analysis of table-format data

A high-level view of the data exploration process was presented in Figure 2.2 (on page 9). The process is highly interactive and iterative, and thus requires considerable amount of manual work. However, many of the data exploration tasks remain the same from one iteration to the next. Figure 5.1 shows a more detailed view which breaks down the analysis phase into several sub-tasks. These are divided along two main tracks: variable and sample analysis.

5.1.1 Sample analysis

The goal of sample analysis is to find natural (or at least sensible) subsets from the data and by characterizing their properties to give to the data miner an idea of the structure and contents of the data manifold.

Sample analysis starts with normalization and quantization of the data.

- Normalization usually means linear scaling of variables as discussed in Section 3.1.3. More generally, it corresponds to selecting a distance metric in the input space. The distance metric sets the viewpoint from which the whole sample analysis looks into the data by defining how important different variables are in the subsequent analysis methods.

- Quantization transforms the continuous input space into a discrete set of regions. Using representatives of these regions instead of the original data reduces the computational complexity of subsequent analysis algorithms (projection, clustering, and local modeling), as well as averages out noise and outliers, both of

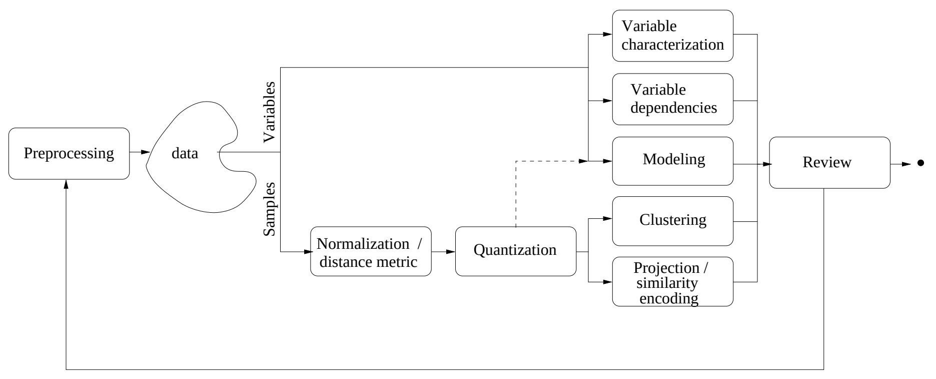


Figure 5.1: Data analysis framework for data exploration of table-format data. The framework consists of a number of blocks corresponding to the core analysis tasks in data exploration. While this thesis concentrates on the use of SOM for data exploration, the blocks could be easily replaced with other similar methods.

which are important aspects in data exploration. Of course, if there is only a small amount of data, the quantization can be discarded.

This is followed by the application of clustering and projection methods to find groups of similar data samples.

- Cluster analysis is at the core of sample analysis by providing quantitative information of the structure and contents of the data set. Even if the data set does not contain natural clusters, it is still useful to partition it to sensible subsets because this allows the data to be investigated at different granularities. Of course, in such a case it would be misleading to imply that there are natural clusters in the data. Therefore, measuring cluster validity is an important part of cluster analysis.
- Projection methods are needed for visualization. Their primary purpose is to provide similarity encodings so that similar samples (in terms of the distance metric) are easy to identify from the visualizations. In practice, at least two projections are needed: a spatial projection and a color coding. The former is more accurate, but cannot be used in visualizations where position information is needed for displaying other information. In such situations, the latter can be used.

5.1.2 Variable analysis

Variable analysis has two main sub-tasks: variable characterization, and variable dependency analysis.

- Characterization of the properties of individual variables is a very fundamental task because knowing the statistics of the data set is essential to be able to detect interesting deviations in its subsets. For characterization, simple descriptive statistics — for example minimum, median, maximum values, standard deviation and a histogram — can be used[1].

[1]In meta-learning, a recently emerged field of research, the aim is to predict the success of different algorithms in a specific classification task based on data set characteristics [60]. This field may, in the future, add

- The goal of dependency analysis is to detect pairs and groups of variables which are strongly related, or have interesting interactions. Dependencies between variables are studied using statistical methods, such as correlation analysis, entropy analysis, factor analysis, Bayesian networks, and rules, see for example [5].

  The methods investigated in this thesis deal with variable dependency analysis mainly through visualizations. In addition, if a suitable presentation for the variables is selected as discussed in Section 4.2.3, clustering and projection techniques can be applied to find groups of related variables.

Often there are one or more output variables in the data set, and their prediction is one of the main goals of the whole data mining process. Therefore, also modeling is a part of exploratory data analysis. However, exploratory modeling is different from the actual modeling phase of the data mining process (see Figure 2.1). Exploratory models are very simple, and rather than being real attempts to solve the modeling problem, they are used to provide an indication of how well different model families might succeed in the modeling task. In general, it is advantageous to try models from different kinds of model families, and validate the performance and reliability of each in order to find out what kinds of models are worth a more detailed investigation[2]. Modeling is tightly linked with dependency analysis, and the results of either task can be utilized to help the other.

5.1.3 Automatically generated report

General requirements

In Publication 10, a report generation system is described which applies the explorative analysis methods discussed above and produces a report of the results autonomously, without user intervention. Of course, exploratory data analysis is an inherently interactive process, and thus cannot be fully automated. Rather, the automated analysis provides an advantageous starting point for the interactive analysis, as shown in Figure 5.2. The automated analysis has also a number of other desirable properties:

- The analysis is easy to repeat.
- The required level of technical know-how of the data miner is reduced when compared to fully interactive data exploration.
- The resulting report acts as documentation which can be later referred to.

The applied analysis methods must cover the tasks discussed in Section 5.1. However, just any set of algorithms is not suitable.

- Because the algorithms are applied autonomously, they must be robust to errors and to missing values. The algorithms need not be perfect in this sense, though. One of the purposes of data exploration is to catch errors from the data. Many fundamental errors in the data are easy to notice from the report, after which the errors can be removed by preprocessing, and the report can be regenerated.

some important items to this list and, in general, shed some valuable new light on the data set characterization problem.

[2]One very good alternative are nearest neighbor based methods, like the local models discussed in Section 4.4. These are known to be robust and to produce relatively good models [56].

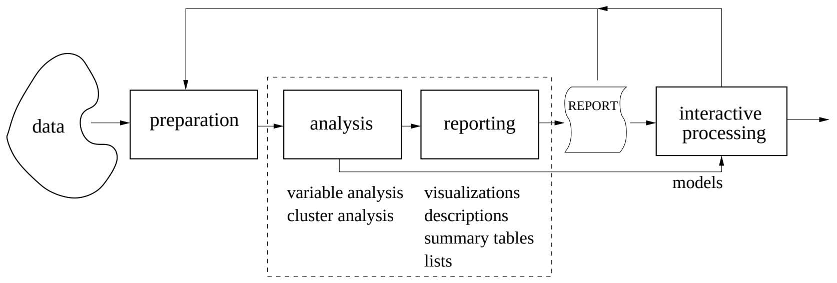


Figure 5.2: Partially automated data exploration process. After preparing the data, the analysis methods are applied autonomously, and a survey report on the findings is produced. Based on the findings in the report and possibly further insights based on interactive investigation, the data miner may either proceed with the next data mining phase, or prepare the data set better and, with a push of a button, make a new report.

- Preferably the methods should not require predefined parameters at all. When parameters cannot be avoided, the methods should be robust to the choice of their values such that some sensible default values can be used in all/most cases.

- The methods must be relatively fast. Not as fast as in a truly interactive analysis environment, where the user excepts to see the results immediately, but fast enough that the analysis can be frequently redone. Actual time limits or performance requirements depend on the amount of data and the data mining environment.

After applying the analysis methods, a report is written which combines their results into a coherent whole. The elements of the report are visualizations and numerical or linguistic descriptions organized into summary tables and lists. Visualizations allow the display of large amounts of detailed information at once, and thus provide an overview of the data which the data miner can investigate at different levels of detail. Summaries, on the other hand, turn the essentially qualitative information in the visualizations into a set of quantitative statistical or verbal statements. They represent the data in a compact form and make it easier to understand and memorize the properties of the data.

A SOM-based analysis

The sample analysis in Publication 10 is based on SOM. The variables are normalized to have unit variance. The quantization is done using SOM. Besides the prototype vectors, the SOM provides a projection of the prototypes vectors onto the map grid. In addition, another spatial projection is calculated using PCA-projection (Figure 4.2), and a color coding of the prototype vectors (Figure 4.7). The clustering is done using distance matrices (see Section 4.3.3). A cluster hierarchy is constructed on top of these base clusters (see Section 4.3.4), and descriptions for them are generated (see Section 4.3.5).

The variables are characterized using histograms and other basic statistics. Variable dependency analysis is based on correlation coefficients, and the correlation coefficient vectors are used to cluster the variables (see Section 4.2.3). Also component planes are used to visualize the variables (see Figure 4.9). The analysis system does not have a modeling component.

The actual report consists of a short overview for a quick look at the major properties of the data, followed by a more detailed description of each variable and cluster.

A limitation of the report is that it is static: it cannot be manipulated, all analysis results must be calculated beforehand, and all analysis results must be inserted into the report. A major enhancement would be to embed the analysis system in an interactive environment, for example like the one in Figure 4.1. This would give the user more control over the visualizations and models, as well as allow the user to request more information on interesting details.

5.2 Analysis of hierarchical data sets

In Publication 9, hierarchical data sets are investigated. Hierarchical data sets consist of (at least) two levels of data. For each higher level item, a varying number of corresponding lower level items exists. This kind of data is fairly common, for example in marker basket analysis. Also the mills data set was originally hierarchical, consisting of one higher-level data table and two lower level tables (see Appendix A).

To provide an analysis which takes both data levels into account, the lower level information must be transferred to the higher level. This can be done by first training SOMs for the lower level data sets. For each higher level item i_h, the corresponding set of lower level items $\{i \,|\, i_h\}$ is collected, and their responses (see Section 4.2.4) in each unit of the corresponding lower level map are measured:

$$[r(\mathbf{x}_i, \mathbf{m}_1), ..., r(\mathbf{x}_i, \mathbf{m}_M)]. \tag{5.1}$$

The sum of responses in each map unit is used to represent the lower level data on the higher level: each map unit corresponds to a certain kind of lower level item, and the responses measure how frequently that kind of items are associated with the corresponding higher level item.

The problem with such a procedure is that since each unit of the lower level map is represented by a new variable, the number of new variables on the higher level becomes easily quite large. Using only a small number of map units on the lower level map is not a very good solution because this severely limits the analysis of the lower level data. Instead, the number of new variables can be reduced using some dimension reduction technique, or clustering. The latter approach is used in Publication 9. The lower level map is first clustered, and the responses of data samples in these clusters are used as the new variables:

$$[r(\mathbf{x}_i, C_1), ..., r(\mathbf{x}_i, C_C)]. \tag{5.2}$$

The main advantage of this approach is that since the clusters in the lower level map can be easily analysed, the new variables have meaningful interpretations. See for example Appendix A, where the lower level clusters of the mills data set are shortly described.

In Figure 5.3, the process of exploring hierarchical data sets is shown. Compared with analysis of unconnected data sets, an additional step of determining the responses of the higher level items on the lower level clusters is needed. However, in the actual data exploration, the same methods and processes as in exploration of a single table of data can be used.

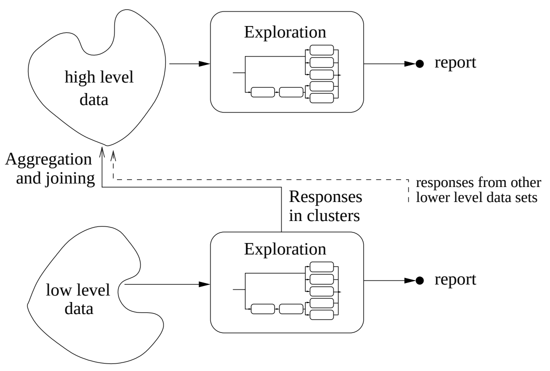


Figure 5.3: Data analysis flow for hierarchical data sets.

5.3 Discussion

A difficulty with the proposed data exploration framework is its validation. This thesis has concentrated on identifying how SOM-based methods can be used in the framework, but it has not been investigated whether they are the best methods for it.

The validity of the presented data exploration process is ultimately determined by the data miner: does it help him / her to achieve better understanding of the data, faster than with alternate exploration processes? This is a complex test situation which depends highly on the prior knowledge of the data miner, quality of the software tools used for exploration, and the nature of the sought knowledge. The data exploration framework should be validated in a usability testing environment, for example as follows.

The test objective would be to compare two data exploration processes to each other. For both processes a set of data mining tools — or for example a set of data reports — would be available. The users would be people who have to deal with data in their work, but have only a limited background knowledge of data mining methods. They would be divided to two sets, one for each data mining tool. Each user would be given some essential training in the use of the tools, and then asked a set of typical data mining questions regarding the given data sets: what kind of groups there are in the data, what kind of dependencies there are, how do two given groups differ from each other, etc. The answers would be evaluated both in terms of truthfulness, and the time required for answering.

Chapter 6

Conclusion

In data mining and knowledge discovery it is essential to understand the data that is being processed. The central task for gaining the necessary understanding is data exploration. This is a highly interactive process in which the data miner iteratively does some data preparation (preprocessing phase), applies suitable analysis methods (analysis phase) and based on the results forms hypothesis (or discards old hypothesis) about the properties of the data (review phase), thus increasing his or her understanding of the data manifold.

In this thesis, a data exploration process based on the Self-Organizing Map (SOM) has been investigated. The basic SOM algorithm has been widely implemented in various software tools and libraries. However, for a common practitioner a difficulty with applying the SOM in data mining has been that there is no wide consensus or understanding of the methods needed for post-processing the SOM. Thus, this work has had two primary goals:

- Firstly, to enhance and investigate the validity of the use of SOM in data exploration. Methods for the two most important application areas of the SOM — visualization and cluster analysis — have been investigated and developed. In addition, local modeling based on SOM has been studied.
- Secondly, to construct a data exploration framework where much of the work can be done without user intervention (see Figure 6.1) and integrate the SOM-based data mining methods to this framework.

In the constructed process framework, the data analysis is divided into several distinct sub-tasks, most notably the analysis of samples and the analysis of variables. The methods investigated in this thesis concentrate mostly on the former. Based on the analysis results, a data survey report describing the most important properties of data manifold is written automatically, without user intervention. The review phase of the data exploration process is then either entirely based on the produced report, or the report provides a starting point for interactive analysis. In such a framework, the attention of the data miner can be directed more toward the actual understanding task, rather than on the application of the analysis methods. The whole data exploration process, in its turn, provides a starting point for the actual model building phase of data mining.

The SOM provides a versatile basis on which to build such an analysis process. It is in key position in providing an initial organization of the data which is useful both in visualization and as a way to keep the projection and clustering methods used in the

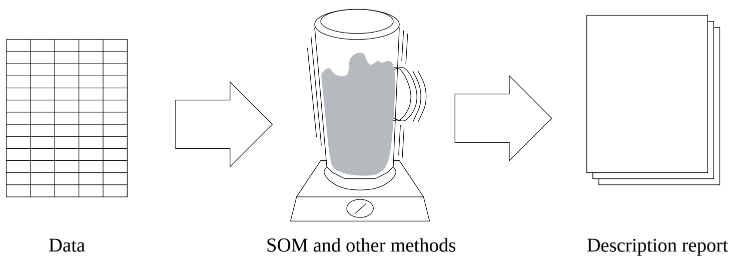


Figure 6.1: Data exploration at the push of a button: a set of algorithms is applied to the given data set and a data survey report is produced without user intervention. The report acts as an overview and a "map" to the data space for the data miner.

analysis computationally feasible. Hopefully, this work provides a convenient reference or starting point for researchers and fellow data engineers to make use of the potential of the SOM as a versatile data mining tool[1].

The key contributions of the exploration process framework — identification of key questions in exploration of table-format data, autonomous application of analysis methods, and integration of different analysis results into a coherent whole — are also key issues in making data mining methods easier to use by the common practitioner. Therefore, I would like to advocate other researchers to consider these issues, extend them, and pave the way for more user-friendly data analysis.

[1]The SOM training algorithm, as well as many of the methods mentioned in this thesis have been implemented in the SOM Toolbox [141] for Matlab computing environment: `http://www.cis.hut.fi/projects/somtoolbox/`

Bibliography

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*, pages 94–105, Seattle, Washington, 1998.

[2] D. Alahakoon and S. K. Halgamuge. Knowledge Discovery with Supervised and Unsupervised Self Evolving Neural Networks. In Yamakawa and Matsumoto [148], pages 907–910.

[3] Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, July 2001.

[4] R.A. Becker and W.S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[5] Michael Berthold and David J. Hand, editors. *Intelligent Data Analysis: an Introduction*. Springer, 1999.

[6] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[7] James C. Bezdek and Sankar K. Pal, editors. *Fuzzy Models for Pattern Recognition: Methods That Search for Structures in Data*. IEEE Press, 1992.

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

[9] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[10] Justine Blackmore and Risto Miikkulainen. Visualizing High-Dimensional Structure with the Incremental Grid Growing Neural Network. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the 12th International Conference*, pages 55–63. Kaufmann, 1995.

[11] Eric Boudaillier and Georges Hebrail. Interactive Interpretation of Hierarchical Clustering. *Intelligent Data Analysis*, 2(3), August 1998.

[12] Ronald J. Brachman and Tej Anand. Advances in knowledge discovery and data mining. In Fayyad et al. [32], chapter 2: The Process of Knowledge Discovery in Databases.

[13] P.S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of 15th International Conference on Machine Learning*, pages 91–99. Morgan Kaufmann, San Francisco, CA, 1998.

[14] Joachim Buhmann. Complexity Optimized Data Clustering by Competitive Neural Networks. *Neural Computation*, 5(3):75–88, May 1993.

[15] Andreas Buja, John Alan McDonald, John Michalak, and Werner Stuetzle. Interactive data visualization using focusing and linking. In *Proceedings of IEEE Conference on Visualization*, pages 156–163, 1991.

[16] Pete Chapman, Julian Clinton, Thomas Khabaza, Thomas Reinartz, and Rüdiger Wirth. CRISP-DM 1.0 step-by-step data mining guide. Technical report, CRISM-DM consortium, 2000. `http://www.crisp-dm.org`.

[17] V. Cherkassky and H. Lari-Najafi. Constrained topological mapping for nonparametric regression analysis. *Neural Networks*, 4:27–40, 1991.

[18] Vladimir Cherkassky and Filip Mulier. *Learning From Data: concepts, theory, and methods*. John Wiley & Sons, Inc., 1998.

[19] Herman Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 63:361–368, 1973.

[20] Krzysztof Cios, Witold Pedrycz, and Roman Swiniarski. *Data Mining Methods for Knowledge Discovery*. Kluwer Academic Publishers, 1998.

[21] M. Cottrell, E. de Bodt, and M. Verleysen. A statistical tool to assess the reliability of self-organizing maps. In Nigel Allinson, Hujun Yin, Lesley Allinson, and Jon Slack, editors, *Advances in Self-Organizing Maps, Proceedings of the Workshop on Self-Organizing Maps 2001*, pages 7–14. Springer, June 2001.

[22] Guido Deboeck and Teuvo Kohonen, editors. *Visual explorations in Finance using Self-Organizing Maps*. Springer-Verlag, London, 1998.

[23] P. Demartines and J. Hérault. Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, January 1997.

[24] Pierre Demartines and Jeanny Hérault. Vector quantization and projection neural network. In A. Pieto, J. Mira, and J. Cabestany, editors, *International Workshop on Artificial Neural Networks (IWANN'93)*, volume 686, pages 328–333. Springer-Verlag, 1993.

[25] R. Der and M. Herrmann. Nonliear chaos control by neural nets. In *Proceedings of the Internatianl Conference on Artificial Neural Networks (ICANN) 94*, 1994.

[26] Inderjit S. Dhillon, Dharmendra S. Modha, and W. Scott Spangler. Visualizing class structure of multidimensional data. In *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics*, Minneapolis, MN, May 1998.

[27] Ignacio Diaz, Alberto B. Diez, and Abel A. Cuadrado Vega. Complex process visualization through continuous feature maps using radial basis functions. In *Proceedings of the International Conference on Artificial Neural Networks*, 2001.

[28] Richard O. Duda and Peter E. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, 1973.

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

[30] Vladimir Estivill-Castro, Ickjai Lee, and Alan T. Murray. Criteria on proximity graphs for boundary extraction and spatial clustering. In Qing Li David Cheung, Graham J. Williams, editor, *Proceedings of the Pacific-Asia Conference Advances in Knowledge Discovery and Data Mining (PAKDD2001)*, pages 348–357. Springer, April 2001.

[31] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Proceeding of The Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 82–88, 1996.

[32] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press / The MIT Press, California, 1996.

[33] Arthut Flexer. Limitations of self-organizing maps for vector quantization and multidimensional scaling. In *Advances in Neural Information Processing Systems (NIPS) 9*, pages 445–451. MIT Press, 1997.

[34] Bernd Fritzke. Let it grow – self-organizing feature maps with problem dependent cell structure. In Kohonen et al. [76], pages 403–408.

[35] Bernd Fritzke. Growing Cell Structures – A Self-Organizing Neural Network for Unsupervised and Supervised Learning. *Neural Networks*, 7(9):1441–1460, 1994.

[36] Bernd Fritzke. Growing Grid – a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995.

[37] Allen Gersho. Asymptotically Optimal Block Quantization. *IEEE Transactions on Information Theory*, IT-25(4):373–380, July 1979.

[38] Thore Graepel, Matthias Burger, and Klaus Obermayer. Phase transitions in stochastic self-organizing maps. *Physical Review E*, 56:3876–3890, 1997.

[39] Robert M. Gray. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[40] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, New York, 1998. ACM.

[41] Erkki Häkkinen and Pasi Koikkalainen. The neural data analysis environment. In *Proceedings of the Workshop on Self-Organizing Map*, pages 69–74, 1997.

[42] A. Hämäläinen. A measure of disorder for the self-organizing map. In *Proceedings of International Conference on Neural Networks (ICNN) 94*, pages 659–664, 1994.

[43] Ari Hämäläinen. *Self-Organizing Map and Reduced Kernel Density Estimation*. PhD thesis, University of Helsinki, 1995.

[44] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: An attribute-oriented approach. In Li-Yan Yuan, editor, *Proceedings of the 18th International Conference on Very Large Databases*, pages 547–559, San Francisco, U.S.A., 1992. Morgan Kaufmann Publishers.

[45] David Hand, Heikki Mannila, and Padraic Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[46] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[47] Christopher G. Healey. *Effective Visualization of Large Multidimensional Datasets*. PhD thesis, The University of British Columbia, September 1996.

[48] Tom M. Heskes and Bert Kappen. Error potential for self-organization. In *Proc. ICNN'93, Int. Conf. on Neural Networks*, volume III, pages 1219–1223, Piscataway, NJ, 1993. IEEE Service Center.

[49] Johan Himberg. Enhancing SOM-based data visualization by linking different data projections. In L. Xu, L. W. Chan, and I. King, editors, *Intelligent Data Engineering and Learning (IDEAL'98)*, pages 427–434. Springer, 1998.

[50] Johan Himberg. A SOM based cluster visualization and its application for false coloring. In *Proceedings of International Joint Conference in Neural Networks (IJCNN) 2000*, Como, Italy, 2000.

[51] David C. Hoaglin, Frederick Mosteller, and John W. Tukey, editors. *Understanding Robust and Exploratory Data Analysis*. John Wiley & Sons, Inc., 2000.

[52] Jaakko Hollmén, Volker Tresp, and Olli Simula. A self-organizing map for clustering probabilistic models. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99)*, volume 2, pages 946–951. IEE, 1999.

[53] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[54] Jukka Iivarinen, Teuvo Kohonen, Jari Kangas, and Sami Kaski. Visualizing the Clusters on the Self-Organizing Map. In Christer Carlsson, Timo Järvi, and Tapio Reponen, editors, *Proceedings of Conference on Artificial Intelligence Research in Finland*, number 12 in Proceedings of Conference of Finnish Artificial Intelligence Society, pages 122–126, Helsinki, Finland, 1994. Finnish Artificial Intelligence Society.

[55] A. Inselberg and B. Dimsday. Parallel coordinates: A tool for visualizing multidimensional geometry. In *Proceedings of IEEE Conference on Visualization*, pages 361–378, Los Angeles, 1990.

[56] John F. Elder IV and Daryl Pregibon. Advances in knowledge discovery and data mining. In Fayyad et al. [32], chapter 4: A Statistical Perspective on Knowledge Discovery in Databases, pages 83–113.

[57] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[58] Stefan Jockusch. A neural network which adapts its stucture to a given set of patterns. In R. Eckmiller, G. Hartmann, and G. Hauske, editors, *Parallel Processing in Neural Systems and Computers*, pages 169–172. Elsevier Science Publishers, 1990.

[59] Jyrki Joutsensalo. Nonlinear data compression and representation by combining self-organizing map and subspace rule. In *Proceedings of the International Conference on Neural Networks (ICNN'94)*, pages 637–640, Piscataway, NJ, 1994.

[60] Alexandros Kalousis and Melanie Hilario. Feature selection for meta-learning. In Qing Li David Cheung, Graham J. Williams, editor, *Proceedings of the Pacific-Asia Conference Advances in Knowledge Discovery and Data Mining (PAKDD2001)*, pages 222–233. Springer, April 2001.

[61] Jari A. Kangas, Teuvo K. Kohonen, and Jorma T. Laaksonen. Variants of Self-Organizing Maps. *IEEE Transactions on Neural Networks*, 1(1):93–99, March 1990.

[62] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical Clustering Using Dynamic Modeling. *IEEE Computer*, 32(8):68–74, August 1999.

[63] S. Kaski and K. Lagus. Comparing self-organizing maps. In *Proceedings of International Conference on Artificial Neural Networks (ICANN) 96*, pages 809 – 814, 1996.

[64] Samuel Kaski. *Data Exploration Using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, 1997. Acta Polytechnica Scandinavica: Mathematics, Computing and Management in Engineering, 82.

[65] Samuel Kaski. Fast winner search for SOM-based monitoring and retrieval of high-dimensional data. In *Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks*, volume 2, pages 940–945. IEE, London, 1999.

[66] Samuel Kaski, Jari Kangas, and Teuvo Kohonen. Bibliography of self-organizing map (SOM) papers: 1981-1997. *Neural Computing Surveys*, 1:102–350, 1998.

[67] Samuel Kaski, Janne Nikkilä, and Teuvo Kohonen. Methods for interpreting a self-organized map in data analysis. In Michel Verleysen, editor, *Proceedings of ESANN'98, 6th European Symposium on Artificial Neural Networks, Bruges, April 22-24*, pages 185–190. D-Facto, Brussels, Belgium, 1998.

[68] Samuel Kaski, Jarkko Venna, and Teuvo Kohonen. 14: Tips for Processing and Color-Coding of Self-Organizing Maps. In Deboeck and Kohonen [22], pages 195–202.

[69] Samuel Kaski, Jarkko Venna, and Teuvo Kohonen. Coloring that reveals high-dimensional structures in data. In T. Gedeon, P. Wong, S. Halgamuge, N. Kasabov, D. Nauck, and K. Fukushima, editors, *Proceedings of ICONIP'99, 6th International Conference on Neural Information Processing*, volume II, pages 729–734. IEEE Service Center, Piscataway, NJ, 1999.

[70] 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 II, pages 1531–1534, Amsterdam, Netherlands, 1992. North-Holland.

[71] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: and Introduction to Cluster Analysis*. John Wiley & Sons, Inc., 1990.

[72] K. Kiviluoto. Comparing 2D and 3D self-organizing maps in financial data visualization. In Yamakawa and Matsumoto [148], pages 68–71.

[73] Kimmo Kiviluoto. Topology preservation in self-organizing maps. In *Proceedings of the International Conference on Neural Networks (ICNN'96)*, volume 1, pages 294–299, Piscataway, New Jersey, USA, June 1996. IEEE Neural Networks Council.

[74] Kimmo Kiviluoto and Erkki Oja. S-map: A network with a simple self-organization algorithm for generative topographic mappings. In *In Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[75] K. Koffka. *Principles of Gestalt Psychology*. Harcourt-Brace, New York, 1935.

[76] T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors. *Artificial Neural Networks*. Elsevier Science Publishers, 1991.

[77] Teuvo Kohonen. Automatic formation of topological maps of patterns in a self-organizing system. In Erkki Oja and Olli Simula, editors, *Proc. 2SCIA, Scand. Conf. on Image Analysis*, pages 214–220, Helsinki, Finland, 1981. Suomen Hahmontunnistustutkimuksen Seura r. y.

[78] Teuvo Kohonen. Self-Organizing Maps: Optimization Approaches. In Kohonen et al. [76], pages 981–990.

[79] Teuvo Kohonen. Comparison of SOM Point Densities Based on Different Criteria. *Neural Computation*, 11(8):2081–2095, 1999.

[80] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 3rd edition, 2001.

[81] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojärvi, Jukka Honkela, Vesa Paatero, and Antti Saarela. Self organization of a massive text document collection. In Erkki Oja and Samuel Kaski, editors, *Kohonen Maps*, pages 171–182. Elsevier, Amsterdam, 1999.

[82] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojärvi, Jukka Honkela, Vesa Paatero, and Antti Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, (11):574–585, 2000.

[83] Teuvo Kohonen, Samuel Kaski, and Harri Lappalainen. Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM. *Neural Computation*, 9:1321–1344, 1997.

[84] Teuvo Kohonen, Erkki Oja, Olli Simula, Ari Visa, and Jari Kangas. Engineering Applications of the Self-Organizing Map. *Proceedings of the IEEE*, 84(10):1358–1384, October 1996.

[85] Teuvo Kohonen and Panu Somervuo. Self-organizing maps of symbol strings. *Neurocomputing*, 21:19–30, 1998.

[86] Pasi Koikkalainen. Fast deterministic self-organizing maps. In *Proceedings of International Conference on Artificial Neural Networks (ICANN'95)*, pages 63–68, 1995.

[87] Andreas König. A survey of methods for multivariate data projection, visualization and interactive analysis. In Yamakawa and Matsumoto [148], pages 55–59.

[88] M. A. Kraaijveld, J. Mao, and A. K. Jain. A nonlinear projection method based on kohonen's topology preserving maps. *IEEE Transactions on Neural Networks*, 6(3):548–559, May 1995.

[89] J. B. Kruskal. Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, 29(1):1–27, March 1964.

[90] Krista Lagus, Esa Alhoniemi, and Harri Valpola. Independent variable group analysis. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science, pages 203–210. Springer, 2001.

[91] Krista Lagus and Samuel Kaski. Keyword selection method for characterizing text document maps. In *Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks*, volume 1, pages 371–376. IEE, London, 1999.

[92] Jouko Lampinen and Timo Kostiainen. Self-organizing map in data-analysis — notes on overfitting and overinterpretation. In *Proceedings of ESANN'2000*, pages 239–244, Bruges, Belgium, April 2000.

[93] Jouko Lampinen and Timo Kostiainen. Generative probability density model in the Self-Organizing Map. In U. Seiffert and L. Jain, editors, *Self-organizing neural networks: Recent advances and applications*, pages 75–94. Physica Verlag, 2002.

[94] Jouko Lampinen and Erkki Oja. Self-organizing maps for spatial and temporal AR models. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA'89)*, pages 120–127, 1989.

[95] Jouko Lampinen and Erkki Oja. Clustering Properties of Hierarchical Self-Organizing Maps. *Journal of Mathematical Imaging and Vision*, 2(2-3):261–272, November 1992.

[96] R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier. A Scalable Parallel Algorithm for Self-Organizing Maps with Applications to Sparse Data Problems. *Data mining and knowledge discovery*, 3(2):171–195, June 199.

[97] R. C. T. Lee, J. R. Slagle, and H. Blum. A Triangulation Method for the Sequential Mapping of Points from N-Space to Two-Space. *IEEE Transactions on Computers*, C-26(3):288–292, March 1977.

[98] S. P. Luttrell. Self-organisation: A derivation from first principles of a class of learning algorithms. In *Proc. IJCNN'89. Int Joint Conf. on Neural Networks*, volume II, pages 495–498, Piscataway, NJ, 1989. IEEE Technical Activities Board, Neural Network Committee, USA; Int Neural Network Soc, IEEE Service Center.

[99] Paul Mangiameli, Shaw K. Chen, and David West. A comparison of SOM Neural Network and hierarchical clustering methods. *European Journal of Operational Research*, 93(2), September 1996.

[100] J. Mao and A.K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transaction on Neural Networks*, 6(2):296–317, March 1995.

[101] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.

[102] Geoffrey J. McLahlan and Kaye E. Basford. *Mixture Models: Inference and Applications to Clustering*, volume 84 of *Statistics: Textbooks and Monographs*. Marcel Dekker, 1987.

[103] D. Merkl and A. Rauber. Alternative ways for cluster visualization in self-organizing maps. In *Proceedings of the Workshop on Self-Organizing Map*, pages 106–111, 1997.

[104] R. S. Michalski and R. Stepp. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:396–410, 1983.

[105] Glenn W. Milligan and Martha C. Cooper. An Examination of Procedures for Determining the Number of Clusters in a Data Set. *Psychometrika*, 50(2):159–179, June 1985.

[106] Glenn W. Milligan and Martha C. Cooper. A study of standardation of variables in cluster analysis. *Journal of Classification*, 5:181–204, 1988.

[107] John Moody and Christian J. Darken. Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, 1(2):281–294, 1989.

[108] F. Murtagh. Interpreting the Kohonen self-organizing map using contiguity-constrained clustering. *Pattern Recognition Letters*, 16:399–408, 1995.

[109] Gregory Piateetsky-Shapiro. The data mining industry coming of age. *IEEE Intelligent Systems*, pages 32–34, 1999.

[110] J. C. Principe and L. Wang. Non-linear time series modeling with self-organization feature maps. In *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V*, 1995.

[111] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.

[112] Andreas Rauber and Dieter Merkl. Automatic labeling of self-organizing maps: Making a treasure-map reveal its secrets. In *Proceedings of the 3rd Pasific-Area Conference on Knowledge Discovery and Data Mining (PAKDD'99)*, 1999.

[113] Marina Resta. Self organizing evolutionary models in financial markets forecasting. pages 187–190. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

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

[115] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps: an Introduction*. Addison-Wesley, Reading, MA, 1988.

[116] Helge Ritter. Asymptotic Level Density for a Class of Vector Quantization Processes. *IEEE Transactions on Neural Networks*, 2(1):173–175, January 1991.

[117] Joaquim S. Rodrigues and Luis B. Almeida. Improving the learning speed in topological maps of pattern. In *Proceeding of International Neural Network Conference*, volume 2, pages 813–816, 1990.

[118] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000.

[119] John W. Sammon, Jr. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[120] Friedhelm Schwenker, Hans Kestler, and Günther Palm. Adaptive clustering and multidimensional scaling of large and highdimensional data sets. In *Proceedings of International Conference on Articifical Neural Networks (ICANN'98)*, volume 2, pages 911–916, 1998.

[121] Wojciech Siedlecki, Kinga Siedlecka, and Jack Sklansky. An overview of mapping techniques for exploratory pattern analysis. *Pattern Recognition*, 21(5):411–429, 1988.

[122] Olli Simula, Esa Alhoniemi, Jaakko Hollmén, and Juha Vesanto. Monitoring and modeling of complex processes using hierarchical self-organizing maps. In *Proceedings of the 1996 IEEE International Symposium on Circuits and Systems*, volume Supplement, pages 73–76. IEEE, May 1996.

[123] Olli Simula and Jari Kangas. *Neural Networks for Chemical Engineers*, volume 6 of *Computer-Aided Chemical Engineering*, chapter 14, Process monitoring and visualization using self-organizing maps. Elsevier, Amsterdam, 1995.

[124] Olli Simula, Juha Vesanto, Petri Vasara, and Riina-Riitta Helminen. *Industrial Applications of Neural Networks (L.C. Jain and V.R. Vemuri, eds.)*, chapter 4: The Self-Organizing Map in Industry Analysis, pages 87–112. CRC Press, 1999.

[125] Janne Sinkkonen and Samuel Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2002.

[126] H. Speckmann, G. Raddatz, and W. Rosenstiel. Considerations of geometrical and fractal dimension of SOM to get better learning results. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'94)*, pages 342–345, 1994.

[127] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, December 2000.

[128] 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, *Proceedings of International Conference on Artificial Neural Networks (ICANN'91)*, pages 847–852, Amsterdam, Netherlands, 1991.

[129] Edward Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[130] J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reading, MA, 1977.

[131] A. Ultsch. Self-organized feature maps for monitoring and knowledge acquisition of a chemical process. In *Proceedings of International Conference on Artificial Neural Networks (ICANN) 1993*, pages 864–867, Amsterdam, September 1993.

[132] A. Ultsch. Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 33–45. Elsevier, 1999.

[133] A. Ultsch, G. Guimaraes, D. Korus, and H. Li. Knowledge extraction from artificial neural networks and applications. In *Proceedings of Transputer-Anwender-Treffen / World-Transputer-Congress (TAT/WTC) 1993*, pages 194–203, Aachen, Tagungsband, September 1993. Springer Verlag.

[134] A. Ultsch and H. P. Siemon. Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis. In *Proceedings of International Neural Network Conference (INNC'90)*, pages 305–308, Dordrecht, Netherlands, 1990. Kluwer.

[135] A. Vellido, P.J.G Lisboa, and K. Meehan. Segmentation of the on-line shopping market using neural networks. *Expert Systems with Applications*, 17:303–314, 1999.

[136] Jarkko Venna and Samuel Kaski. Neighborhood preservation in nonlinear projection methods: an experimental study. In *Proceedings of International Conference on Artificial Neural Networks (ICANN) 2001*, 2001.

[137] Juha Vesanto. Data mining techniques based on the self-organizing map. Master's thesis, Helsinki University of Technology, 1997.

[138] Juha Vesanto. Neural network tool for data mining: SOM Toolbox. In *Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000)*, pages 184–196, Oulu, Finland, 2000. Oulun yliopistopaino.

[139] Juha Vesanto and Jussi Ahola. Hunting for Correlations in Data Using the Self-Organizing Map. pages 279–285. ICSC Academic Press, 1999.

[140] Juha Vesanto and Esa Alhoniemi. Clustering of the Self-Organizing Map. *IEEE Transactions on Neural Networks*, 11(2):586–600, March 2000.

[141] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Self-organizing map in matlab: the SOM toolbox. In *Proceedings of the Matlab DSP Conference 1999*, pages 35–40, Espoo, Finland, November 1999.

[142] Juha Vesanto, Johan Himberg, Markus Siponen, and Olli Simula. Enhancing SOM Based Data Visualization. In Yamakawa and Matsumoto [148], pages 64–67.

[143] Juha Vesanto, Petri Vasara, Riina-Riitta Helminen, and Olli Simula. Integrating environmental, technological and financial data in forest industry analysis. In Bert Kappen and Stan Gielen, editors, *Proceedings of 1997 Stichting Neurale Netwerke Conference*, pages 153–156, Ansterdam, the Netherlands, May 1997. Stichting Neurale Netwerke, University of Nihmegen, World Scientific.

[144] T. Villmann, R. Der, M. Herrmann, and T.M. Martinetz. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, 1997.

[145] Th. Villmann, R. Der, and Th. Martinetz. A new quatitative measure of topology preservation in kohonen's feature maps. In *Proc of International Conference on Neural Networks (ICNN) 94*, pages 645–648, 1994.

[146] J. Walter, H. Ritter, and K. Schulten. Non-linear prediction with self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN) 90*, 1990.

[147] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[148] T. Yamakawa and G. Matsumoto, editors. *Proceedings of the 5th International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA'98)*. World Scientific, 1998.

[149] Paul L. Zador. Asymptotic Quantization Error of Continuous Signals and the Quantization Dimension. *IEEE Transactions on Information Theory*, IT-28(2):139–149, March 1982.

[150] Tian Zhang, Raghu Ramakrishnan, , and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, 1996.

[151] Xuegong Zhang and Yanda Li. Self-Organizing Map as a New Method for Clustering and Data Analysis. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'93)*, pages 2448–2451, 1993.

Appendix A

Data sets used in examples

System data

The system data is a relatively simple 9-dimensional data set measuring the disk, CPU and network performance of a workstation in a network environment. Four of the variables reflect the volumes of network traffic and five of them the CPU usage in relative measures. The data forms a 9-dimensional time-series, but in this case it is considered as independent samples from a system. One of the central tasks would be, based on the measurements, to monitor the state of the system in real-time.

The operation of a single computer workstation in a networking environment was recorded in terms of the central processing unit (CPU) activity and the volumes of network traffic. The recordings were made during 5 working days with intervals of 2 minutes and 5 minutes, during day operations and night time, respectively. The number of samples is 1908. The data was collected by Dr. Jaakko Hollmén in 1996 in Laboratory of Computer and Information Science, Helsinki University of Technology.

Variable	**Explanation**
`blks/s`	read blocks per second
`wblks/s`	written blocks per second
`ipkts`	the number of input packets
`opkts`	the number of output packets
`usr`	time spent in user processes
`sys`	time spent in system processes
`intr`	time spent handling interrupts
`wio`	CPU was idle while waiting for I/O
`idle`	CPU was idle and not waiting for anything

Mills data

Original tables

The mills data is the pulp and paper mill data set that was the original inspiration for this work. It was provided by Jaakko Pöyry Consulting. The data contained information of the technology of the pulp and paper mills around the world. It was divided into three separate sets: information of the mill itself, its paper machines and its pulp lines. Each mill may have zero, one or several paper and pulp lines. There were altogether 4205 pulp and paper mills, with 8765 paper machines and 2979 pulp lines in them. The variables in each table are listed below.

Mill-level data

Variable	Explanation
TOT_PAPER	total paper production capacity (1000 t/a)
NO_PMS	number of paper machines
COATERS	number of coaters
NEWS	newsprint production capacity (% of total paper production capacity)
TOT_PR_WR	print/write paper production capacity (%)
TOT_WF	woodfree (WF) paper production capacity (%)
UNC_WF	uncoated WF paper production capacity (%)
CTD_WF	coated WF paper production capacity (%)
TOT_WC	wood containing (WC) paper production capacity (%)
UNC_WC	uncoated WC paper production capacity (%)
CTD_WC	coated WC paper production capacity (%)
TOT_IND	industial papers production capacity (%)
WRAPP	wrapping paper production capacity (%)
CARTONB	cartonboard production capacity (%)
LINERB	linerboard production capacity (%)
FLUTING	fluting production capacity (%)
TISSUE	tissue production capacity (%)
OTHER	other papers production capacity (%)
TOT_PULP	total pulp production (1000 t/a)
TOT_CHEM	chemical pulp production capacity (%)
UBL_SA	unbleached sulphate (Sa) production capacity (%)
SBL_SA	semibleached Sa production capacity (%)
BL_SA	bleached Sa production capacity (%)
UBL_SI	unbleached sulphite (Si) production capacity (%)
SBL_SI	semibleached Si production capacity (%)
BL_SI	bleached Si production capacity (%)
TOT_CMECH	chemimechanical pulp production capacity (%)
CTMP_TOT	chemi-thermomechanical pulp (CTmp) production capacity (%)
TOT_MECH	mechanical pulp production capacity (%)
GW	ground wood production capacity (%)
RMP	refiner mechanical pulp (Rmp) production capacity (%)
TMP	thermomechanical pulp (Tmp) production capacity (%)
DEWA	deinked waste paper (Dewa) production capacity (%)
DIWA	disperged waste paper (Diwa) production capacity (%)
SEMI_CHEM	semichemical pulp production capacity (%)

Pulp line data

Variable	Explanation
BLEACHING	bleaching type
FIBRE	fibre type
MAINGRADE	main pulp grade type
MARKETCAP	market pulp capacity (%)
TOTALCAP	total capacity (1000 t/a)

Paper machine data

Variable	Explanation
`WIRE`	wire width (mm)
`WTRIM`	wire trim width (mm)
`SPEEDDES`	speed (m/min)
`GRAMMIN`	grammage, min (g/m^2)
`GRAMMAX`	grammage, max (g/m^2)
`CAPACITY`	capacity (1000 t/a)
`BUILT`	(re)built (year)

Handling lower-level tables

The information in the two lower-level data tables (paper machines and pulp lines) had to be transferred to the upper level. This was done using clustering approach introduced in Section 5.2. The responses were defined simply as

$$\mathbf{r}_i = [\delta(1, c(b_i)), ..., \delta(C, c(b_i))],$$

where $c(b_i)$ is the cluster index of map unit b_i and $\delta(\cdot,\cdot)$ is the Dirac delta function. The clustering was done automatically using the approach detailed in Section 4.3.3. The number of clusters on the paper machine map and pulp line maps were 9 and 15, respectively. Combined with the original capacity information, the total number of variables variables on the mills data is 59. The new components corresponding to the clusters on the lower-level maps are listed below.

Variable	Explanation
`Paper:Small1`	Small paper machines with low speed.
`Paper:Smallpaper2`	Like `Paper:Small1`.
`Paper:NewSmall1`	Small capacity, lightweight paper, but new.
`Paper:NewSmall2`	Small machines, but new.
`Paper:SpeedWide`	Wide trim, and high speed.
`Paper:Speed`	High speed, but narrow trim.
`Paper:Gramm`	Heavy papers.
`Paper:Old`	Old machines.
`Paper:Big`	High capacity, wide trim.

Variable	Explanation
`Pulp:BleachedWood1`	Chemically produced bleached pulp from wood.
`Pulp:BleachedWood2`	Like `Pulp:BleachedWood1`.
`Pulp:Totalcapacity`	High capacity pulping lines (chemically from wood).
`Pulp:BleachedMech`	Mechanically produced bleached pulp.
`Pulp:Semibleached`	Semibleached pulp.
`Pulp:Unbleachedmech`	Mechanically produced unbleached pulp.
`Pulp:Waste`	Pulp from waste paper.
`Pulp:Rags`	Pulp from rags.
`Pulp:SemichemicalWood`	Pulp semichemically from wood.
`Pulp:UnbleachedChem`	Unbleached pulp produced chemically.
`Pulp:Otherfibre`	Pulp from other fibres.
`Pulp:Marketcap`	Market capacity high (pulp produced for selling).
`Pulp:OtherfibreUnbleached`	Unbleached pulp from other fibres.
`Pulp:SemichemicalOtherfibre`	Semichemically produced pulp from other fibres.
`Pulp:WasteUnbleached`	Unbleached pulp from waste paper.

Appendix B

SOM-algorithm in C

In C programming language, one epoch of the basic sequential training algorithm for SOM can be written as:

```
for (i=0; i<N; i++) {              /* go through the data: O(N) */
  bmu=-1; min=1000000;
  for (j=0; j<M; j++) {            /* find the BMU: O(3Md) */
    dist=0;
    for (k=0; k<d; k++) { dx = X[i][k] - M[j][k]; dist += dx*dx; }
    if (dist<min) { min=dist; bmu=j; }
  }
  for (j=0; j<M; j++) {            /* update: O(3M+3Md) */
    h = alpha*exp(delta(bmu,j)/rad); /* Gaussian neighborhood */
    for (k=0; k<d; k++) M(j,k) -= h*(M[j][k] - X[i][k]);
  }
}
/* TOTAL: O(6NMd + 3NM) */
```

Above, `X[i][k]` is the *k*th component of the *i*th data sample, `M[j][k]` is the *k*th component of map unit j, `delta` is a table of squared map grid distances $\|\mathbf{r}_{b_i} - \mathbf{r}_j\|^2$ between map units calculated beforehand and `rad` the neighborhood radius at time t multiplied by -2 (see Eq. 3.3). The complexities in the comments (for example `O(N)`) refer to the number of floating point operations (additions, multiplications and exponents).