HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Juha Vesanto

Using SOM in Data Mining

Licentiate's thesis

Thesis for the degree of Licentiate of Science in Technology
Supervisors: Professor Olli Simula
Professor Samuel Kaski

Espoo, Finland 17th April 2000

HELSINKI UNIVERSITY OF TECHNOLOGY	**ABSTRACT OF THE LICENTIATE'S THESIS**
Author:	Juha Vesanto
Title:	Using SOM in Data Mining
Date:	5th April 2000
Number of pages:	49
Department:	Department of Computer Science and Engineering
Professorship:	Tik-115, Information Sciences
Supervisors:	Professor Olli Simula, Professor Samuel Kaski

Data mining as a research area answers to the challenge of analysing large databases in commerce, industry, and research. The purpose is to find new knowledge from databases where the dimensionality, complexity, or amount of data is prohibitively large for manual analysis. Data mining is an interactive process requiring that the intuition and background knowledge of application experts are coupled with the computational efficiency of modern computer technology.

The Self-Organizing Map (SOM) is one of the most popular neural network models. The SOM quantizes the data space formed by the training data and simultaniously performs a topology-preserving projection of the data onto a regular low-dimensional grid. The grid can be used efficiently in visualization.

This thesis consists of an introduction and three publications. In the introduction, an overview of each step of the data mining process is first presented, primarily based on the CRoss-Industry Standard Process model for Data Mining (CRISP-DM). Then the SOM algorithm and some of its variants are introduced, and the use of SOM in data mining is discussed. The publications deal with modeling, visualization and clustering of data using the SOM. In addition, the introduction discusses the use of SOM in summarization.

The SOM is especially suitable for data understanding, but it is a robust tool suitable for modeling and preparation of data as well. It offers a convenient workbench which helps in gaining an initial understanding of the data at hand, and it can be used for creating some initial models as well.

Keywords: Self-Organizing Map, data mining, knowledge discovery in databases, visualization, clustering, summarization, data survey

TEKNILLINEN KORKEAKOULU	**LISENSIAATTITYÖN TIIVISTELMÄ**

Tekijä:	Juha Vesanto
Otsikko:	Itseorganisoivan kartan käyttö tiedonlouhinnassa
Päivämäärä:	5. huhtikuuta, 2000
Sivumäärä:	49
Osasto:	Tietotekniikan osasto
Professuuri:	Tik-115, Informaatiotekniikka
Valvojat:	Professori Olli Simula, Professori Samuel Kaski

Tiedonlouhinta on tieteenala, joka vastaa yrityselämässä, teollisuudessa ja tieteessä kerättyjen suurien tietoaineistojen aiheuttamiin analysointihaasteisiin. Tavoitteena on löytää uutta tietämystä tietokannoista, joiden manuaalinen käsittely ja analyysi on mahdotonta niiden laajuuden tai monimutkaisuuden vuoksi. Tiedonluohinta on interaktiivinen prosessi, jossa tietokoneiden laskennallinen tehokkuus ja ihmisen ymmärrys ongelma-alueesta täytyy onnistuneesti yhdistää.

Itseorganisoituva kartta eli SOM on yksi eniten käytetyistä neuroverkkomalleista. SOM kvantisoi sille annetun tietoaineiston kattaman tietoavaruuden osan ja samanaikaisesti muodostaa topologian säilyttävän kuvauksen tietoavaruudesta kaksiulotteiselle säännölliselle hilalle, jota voidaan käyttää tehokkaasti tiedon visualisointiin.

Tämä opinnäytetyö koostuu johdannosta ja kolmesta julkaisusta. Johdannossa esitetään ensin yleiskuvaus tiedonlouhintaprosessista, perustuen lähinnä CRoss-Industry Standard Process model for Data Mining (CRISP-DM) malliin. Tämän jälkeen esitellään itseorganisoiva kartta ja kuinka sitä voidaan käyttää tiedonlouhinnan eri vaiheissa. Julkaisuissa käsitellään SOMin käyttöä mallinnuksessa, visualisoinnissa ja klusteroinnissa. Näiden lisäksi johdannossa esitellään itseorganisoivan kartan käyttöä luotaessa yhteenvetoa tietoaineistosta.

Itseorganisoiva kartta soveltuu erityisen hyvin tietoaineiston alustavaan tarkasteluun, mutta se on vankka työkalu myös mallinnuksessa ja esikäsittelyssä. Itseorganisoiva kartta toimii hyvänä työpenkkinä, jonka avulla tietoaineistosta voidaan muodostaa yleiskäsitys ja luoda alustavia malleja.

Avainsanat: itseorganisoiva kartta, tiedonlouhinta, tietämyksen muodostaminen tietokannoista, visualisointi, klusterointi

Preface

This licentiate's thesis has been done in the Laboratory of Computer and Information Science in the Helsinki University of Technology, in 1997–2000.

I most sincerely thank my supervisors Professors Olli Simula and Samuel Kaski, as well as Academy Professor Teuvo Kohonen, for their guidance and comments regarding this thesis and the associated publications. Also invaluable has been the support and the spirit of comradeship of the Intelligent Data Engineering research group: Esa Alhoniemi, Jussi Ahola, Johan Himberg, Sampsa Laine, Jukka Parviainen, Juha Parhankangas, Mika Pollari, Pekka Hippeläinen and Professor Olli Simula. 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, and the EU financed Brite/Euram project 'Application of Neural Network Based Models for Optimization of the Rolling Process' (NEUROLL). 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 (Tekniikan edistämissäätiö) and the Finnish Foundation for Commercial and Technical Sciences (Kaupallisten ja teknillisten tieteiden tukisäätiö) is gratefully acknowledged.

Most of all, I would like to dedicate this thesis to my dear wife Riikka and lovely daughter Iida for making the world such a beautiful place. Special thanks to you, i miei soli.

In Otaniemi, 5th April 2000

Juha Vesanto

List of publications and contributions of the author

This thesis consists of an introduction and the following publications:

Publication 1. Esa Alhoniemi, Johan Himberg and Juha Vesanto (1999, June). 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.

This paper 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. The density estimates were compared with those produced by Gaussian Mixture Model (GMM) and by S-Map algorithm. The GMM produced best results, but the estimates based on SOM proved to be good enough to be applied for the intended purpose. The author was responsible for the implementation of S-Map algorithm, but contributed also to other parts of the publication.

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

In this paper, an overview of SOM-based visualization techniques is presented, along with a few new techniques. The SOM is found to be a versatile tool for data visualization. The techniques are divided to three categories: visualization of clusters and the shape of the data manifold, visualization of the variables, and visualization of data on the map. It is shown how different visualizations can be linked together so that their interpretations can be combined.

Unfortunately some references were missing from the original publication (from Tables 1 and 2). An Errata page can be found after the publication. In addition, the figures of the publication have been reprinted.

Publication 3. Juha Vesanto and Esa Alhoniemi (2000). Clustering of the Self-Organizing Map. Accepted for publication in *IEEE Transactions on Neural Networks.*

This paper discusses clustering of the SOM: why it is important and how it can be done. In the experiments, direct clustering of the data was compared with clustering based on SOM and the correspondence was found to be acceptable. The author was responsible for agglomerative clustering of the SOM and comparison between direct and two-level clustering approaches, but contributed also to other parts of the publication.

The publication has not been printed, yet. This version is the final submitted version of the paper.

Abbreviations and notations

CRISP-DM	CRoss Industry Standard Process model for Data Mining
PIE	Prepared Information Environment
BMU	Best Matching Unit
KDD	Knowledge Discovery in Databases
LVQ	Learning Vector Quantizer
pdf	Probability Density Function
SOM	Self-Organizing Map
MDS	Multi-Dimensional Scaling

M	the number of map units / prototypes
N	the number of data samples
d	input space dimension
$\mathbf{x}$	a vector in the input space
$\mathbf{x}_j$	a sample vector j from the input data set
x_{jk}	kth component of sample vector j
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
$\mathbf{m}_i$	reference vector of neuron i
$\mathbf{m}_b$	the best-matching reference vector of an input vector
$\mathbf{r}_i$	location of neuron i in the output space
h_{bi}	neighborhood kernel centered on unit b and evaluated at unit i
V_i	the Voronoi region around unit i
$\delta(t)$	neighborhood radius at time t
$\alpha(t)$	learning rate at time t
$p(\mathbf{x})$	probability density function of random variable $\mathbf{x}$

Contents

Chapter 1

Introduction

Data mining is about finding answers from data. To do this, a wide array of methods is used ranging from relational learning to statistics and neural networks. Strictly speaking, data mining is just a part of Knowledge Discovery in Databases (KDD) [27], but often (and in this thesis as well) the two terms are used synonymously. The purpose of KDD is to find new 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.

The Self-Organizing Map (SOM) [58] is a neural network algorithm that is based on unsupervised learning. It has properties of both vector quantization and vector projection algorithms. The SOM has proven to be a valuable tool in data mining and KDD with applications in full-text and financial data analysis [42, 70, 20]. It has also been successfully applied in various engineering applications in pattern recognition, image analysis, process monitoring and fault diagnosis [4, 62, 98, 99, 103].

In previous work, different SOM-based techniques used for data mining were investigated, and a software tool for using SOM in data mining was presented [108]. In the case study — analysis of the pulp and paper industry of the world [116, 100, 101] — the SOM proved to be an efficient tool for visualization. However, there was a clear lack of post-processing methods, i.e. tools for quantitative analysis of the resulting SOM. Since then, much research has been done in the area.

The purpose of this thesis is to discuss ways to make use of the SOM in data mining. In the second chapter, the different phases of data mining are shortly discussed and a framework called preparation-survey –cycle is introduced. In the third chapter the SOM algorithm and some of its variants are introduced, and the relation of SOM to other widely used algorithms is discussed. In the fourth chapter, the application of the SOM in different data mining tasks is discussed, with special attention given to data understanding.

Data model: unordered table-format data

In this thesis, the table-format data model is assumed: the number of samples may vary, but the number, type and meaning of specific variables in the sample

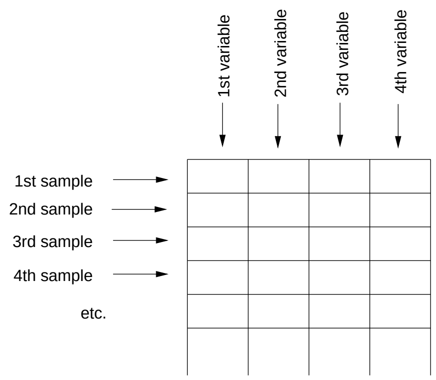


Figure 1.1: Table-format data: there can be any number of samples, but all samples have fixed length, and consist of the same variables.

are known beforehand, see Fig. 1.1. This is because many modeling and analysis algorithms, including the SOM, assume such data model (as opposed to relational data model used by many databases). Not all data mining problems have such simple data sets, but usually the problem can be divided to smaller problems, or the data transformed, such that the methodology is applicable [87]. To use the SOM, the variables should also be numerical, although some analysis of symbolic variables can be performed as well. However, the samples are allowed to have missing values [94].

The thesis mainly concerns analysis of unordered samples. In ordered data sets the samples have a specific order or position on a special dimension, for example time. In analysis this dimension has an especially important role: close-by samples on this dimension are considered very closely related, and major attention is given to variations in this local neighborhood. Familiar examples of ordered dats sets are time-series and time-sequences. Also images are ordered data sets: each pixel has a specific (2-dimensional) position, quite apart from the value of the pixel, which needs to be taken into account. While ordered data sets are very important, the case of unordered samples is still more fundamental. The methods applicable to unordered samples can also be applied to ordered samples.

Chapter 2

Data mining

Data mining is essentially a problem-driven process: there is a question, a problem, that needs an answer, a solution. The answer is generally sought from the data. Of course, finding answers from data is an old research subject with roots firmly in statistics. As an independent research area data mining arose in the 1990's, and as a recognized industry it is only now beginning to be established [85].

In the past, 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 late 1980's and especially in 1990'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.

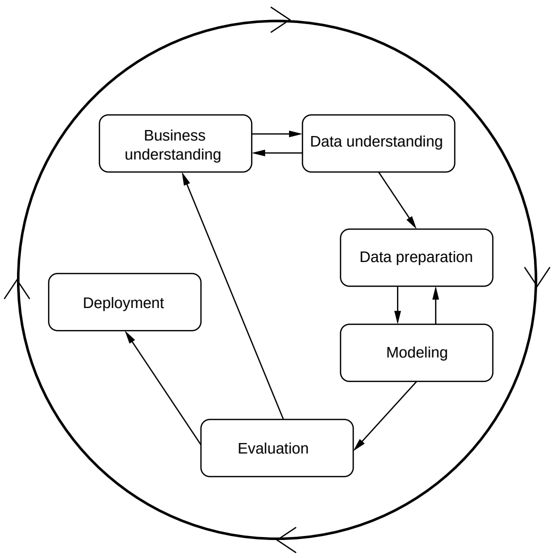
To facilitate the data mining process, there have been attempts to formalize it, especially the CRoss Industry Standard Process model for Data Mining (CRISP-DM) [16] illustrated in Fig. 2.1(a). The data mining process is a cycle having several distinct steps, and a whole lot of backward loops. In CRISP-DM the steps are business understanding, data understanding, data preparation, modeling, evaluation and deployment. The backward loops exist because any step of the data mining process can produce new insights that enable any of the previous steps to be done better. This is important because the first steps form the foundation for the later steps: if they are not done well, the later steps only become that much harder.

In Figs 2.1(b) and (c) are two other process models for data mining [12, 87]. Although the names are slightly different the same overall ideas are present in all three process models: first the problem and data are defined, then the data is prepared and models are built and evaluated. Finally, the new knowledge is consolidated and deployed to solve the problem.

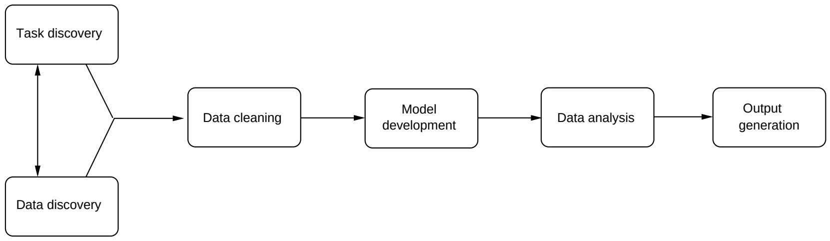
The modeling step is central in data mining because the solution to the problem is basically specified at that point. The solution may be a predictive model of form

$$y = f(\mathbf{x}, \mathbf{w}) \tag{2.1}$$

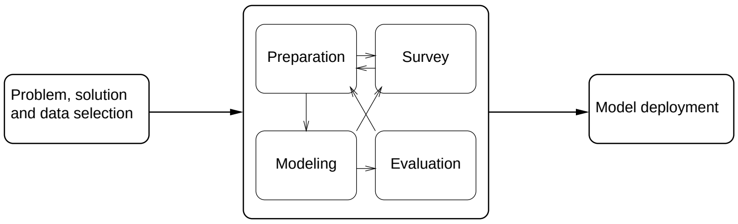
where $\mathbf{x}$ is input data, $\mathbf{w}$ are model parameters and y is the output: continuous in case of a regression problem, one from a small set of possible output values



(a)



(b)



(c)

Figure 2.1: (a) CRISP-DM: process model for data mining, (b) Brachman's KDD process and (c) Pyle's model building outline. Both of the latter process models have been simplified.

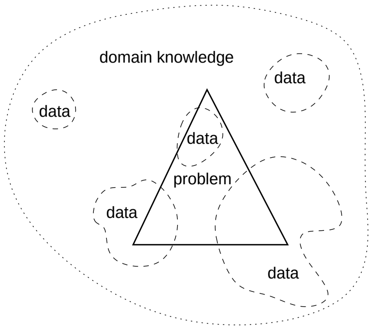


Figure 2.2: Relation of domain knowledge, data and the problem.

in case of a classification problem [17]. Besides these, the desired solution may be more descriptive in nature: a segmentation or clustering of the data into a set of distinct classes, analysis of certain properties like dependencies between variables, or an estimate of the probability density function of the data. These latter solution types are more exploratory in nature and are typically desirable in the early iterations of data mining. In a very general sense a model is something that indicates the essential features of the modelled object, and thus the descriptive solutions can also be called models.

In data mining the models are usually learned from data [17], as opposed to first principle models, where the behaviour of the object is predicted using known laws of nature or other domain knowledge. Domain knowledge can, and should, be used in the data mining process, but data is used at least to tune the parameters of the solution. The success of data mining process depends highly on the quality of the data.

In this chapter, the data mining process is overviewed using the CRISP-DM model as the basis [16]. Also [87] discusses the general characteristics of the data mining process, although it mainly concentrates on data preparation, see Section 2.3. Data modeling is handled very briefly in Section 2.4, although it is perhaps the most throughoutly discussed issue in the literature, see for example [8, 17, 19, 24, 90]. The chapter ends with presentation of preparation-survey –cycle, an important aspect of data mining.

2.1 Business understanding

Data plays a central role in the data mining process. However, raw data — piles and piles of simple measurements, transaction records or log entries — is never self-sufficient. Fig. 2.2 illustrates this: domain knowledge, or business understanding, is needed to glue everything together.

Understanding the problem domain is important in all phases of data mining. The miner is exposed to huge amounts of information in the form of visualizations and statistical characterizations (not to mention the raw data itself). He or she has to make a lot of decisions about what to ignore and what to pursue further, and how. This is impossible without appropriate knowledge of what is interesting, suprising, or relevant with respect to the problem that is being solved. Without the necessary knowledge, the miner is just trashing around in the dark.

The business understanding step has three basic goals:

- To exchange essential *understanding* of the problem domain and of the possibilities of data mining between the project participants. In the ideal case, the person doing the data mining is an expert both in the problem domain and the data mining methods. In such a case, it is relatively clear what questions to ask and which methods can be applied to get answers to them. Typically, though, the people that have the domain knowledge are different from those that have intimate knowledge of the data mining methods. In such a case, the best results are achieved through the extensive involvement and interaction of both parties.

- To define the *problem* first in business terms and then as a specific data mining goal. Ultimately, it is important to arrive at the right problem to solve. Pursuing answers to wrong questions — by having the wrong kind of data or finding an unimplementable solution — is a big danger if the miner does not know anything about the business problem underlying the data mining task. A good problem has a useful and profitable solution, and is solvable with the available data. The right problem is the one that really answers the needs of the customer. In practice, defining the right problem may take a few data mining iterations but even then starting with a good problem is essential to motivate all the associated parties.

- To define the form of the desired *solution*. The desired solution — predictor/classifier used by process engineers, a report or presentation to the maintainance staff, a mathematical model for the research department, a (set of) computer program(s) used by an expert at his workstation — guides the choice of data mining tools and the emphasis put on each. In addition to the form of the solution, some thought should be given to how it is going to be evalutated and finally deployed. If the implementation is unusable, the whole data mining project comes to naught.

While the solution to the defined problem is the primary output of the data mining process, it is not the only one. The process also produces new insights and ideas of the problem domain and related issues. However, these insights are mainly acquired by the miner — the one doing the data mining — not by the receiving party who only gets the deliverable. Also for this reason participation and interactivity between the project partners is very important.

2.2 Data understanding

In CRISP-DM data understanding step is primarily concerned with data collection and, as part of this, understanding the meta-structure of the data: what data is available, what is its source, form, structure and reliability. In data mining, the data often has to be taken “as is”: it has probably been gathered, formatted and stored for purposes quite apart from data mining[1]. Understand-

[1] As opposed to experimental procedures adopted in classical statistics where the variables to be measured and the parameter values used are chosen beforehand by the researcher such that they are maximally informative with respect to the original problem.

ing this purpose and the steps taken to reach it is very important, because such procedures may introduce considerable bias into the data. The following questions may be helpful:

- What is the data set hierarchy? If there are multiple data sets, how are they linked to each other?
- What is the reliability of different measurements? Some measurements may be correct at certain times, while totally unreliable at others. It may be that it is better to reject certain measurements completely than to bring their erroneous, possibly dangerously biased information to the analysis.
- If something fundamental changes in the process along the way, data before and after the change may be completely incomparable. In order to get reliable analysis results, these kinds of data patches must be handled separately.
- If aggregations (combinations of several data items into a single aggregate value, for example median) are necessary, what kind of aggregations would make most sense in solving the problem? What do the aggregates mean in practice?

The driving question of data collection is: can the data possibly solve the given problem? Understanding the data gives a preliminary answer to this question, and thus it guides data collection. It is also needed to be able to preprocess the data appropriately, and to know when the models are reliable. In addition to knowing the meta-structure of the data it is important to get a feel of the data manifold: the structure and shape of the "cloud" formed by the data set.

- What are the different values of the variables and their distribution? Which values are (a)typical? Which of the values are important and which are errors?
- What kind of values appear together? Because human eye is a very sophisticated general-purpose pattern recognition system simple visualizations like scatter plots and time-series plots are important tools.

Actually, understanding the data manifold is a task which has to be undertaken again and again. This aspect of data mining is discussed in more detail in Section 2.7.

Apart from the raw data, one also has to collect all necessary or useful information of the problem: results of previous tests, technical reports, process charts and theoretical knowledge. A good source of information are, of course, the domain experts. The data miner should interview them for hints of:

- what to look at,
- what to ignore,
- what is the valid and interesting range of results / measurement values and
- what kind of solution they wish for or would be satisfied with.

Naturally, just any data isn't good. A good data set holds information relevant to the problem, and little else. If possible, a priori knowledge should be used to select important variables for the mining process. However, if uncertain is it better to take too many variables than too few. It is part of the data mining process to weed out the unimportant or unreliable variables or to device preparation methods that create relevant variables from the existing ones.

In addition to variables, the miner may have to leave out certain samples. Data selection and/or sampling are necessary if there is so much data that using all of it is not computationally feasible, or if there are irrevocably erroneous samples, or if the data set is biased in such a way that it makes modeling difficult or impossible. For example, consider the task of modeling faults in a machine in operative use: there are so few samples of faulty situations that they are lost in the sea of samples from normal operation, and because of this the modeling tool silently ignores them. In this case it would make sense to balance the distribution by leaving out most measurements of normal operation.

Since data is one of the core components of data mining, it should be readily available at the start of the project. In practice, though, data collection (or waiting for it) may be one of the most time-consuming parts of data mining. The original data set may be insufficient, and one has to return to collect more data. The data may have to be collected from a number of sources, or from a source that is only available by special request.

2.3 Data preparation

Data preparation is a diverse and difficult issue. It is so application dependent that it is impossible to give a universally valid specification of how it should be done. However, some general guidelines can be given [87].

The fundamental aim of data preparation is to make it easier to build precise and reliable models. There exists a wide array of different algorithms for constructing models from data each with their own strengths and weaknesses. While some modeling methods are universal approximators being able to capture extremely complex interactions in the data, they can only do this given enough data and computing time. What is more, the models are generic in nature and cannot utilize domain knowledge. Only the human data miner can transform the diverse and often vague domain knowledge into specific preparation procedures which make essential information readily utilizable by the modeling tool. Succesful preparation enables one to construct more reliable and more understandable models faster and with less data.

The preparation step has several aims:

- To select variables and data sets to be used for building the models.
- To clean erroneous or otherwise uninteresting values from the data.
- To generate new features which capture interesting problem characteristics better than the original, raw data.
- To transform the data into a format which the modeling tool can best utilize.
- To define (and implement) a prepared information environment.

2.3.1 Data cleaning

Sometimes the available data is error-free. There are no typing errors, measurement failures or other possible sources of clear-cut errors. But typically the data has erroneous values which have to be corrected or removed from the data. Some modeling tools are more robust with respect to erroneous values than others, but also their performance is degraded by errors.

Because the erroneous values are often very different from the rest — i.e. they are outliers — their impact on many modeling tools may be considerable. Fortunately, this makes it easy to find them. In symbolic data the erroneous values may often be detected by their having only a few instances, while the other values have hundreds of instances.

There are several ways to deal with samples with erroneous values. The harshest option is to totally remove the sample in question from the data set. This may, however, considerably diminish the size of data sets. In addition the amount of information about the other variables is lost. Note that the presence of erroneous values might not be completely random. What if in the presence of a high value in one variable (e.g. voltage) the measurement device for another variable breaks down and produces an erroneous value? By ignoring these samples completely this information is lost. A better approach is to replace the erroneous value either with a real, estimated value, or with a "missing" value. The latter, however, is only applicable in case of those modeling tools that can handle partial data.

Another target for data cleaning is noise. Noise, as opposed to simple errors, is something that is an inherent part of the signal. It exists due to the random nature of the world: in continuous real-world measurements there is always some variation around the expected value. Noise, like errors, is uninteresting from the perspective of data mining. Therefore, as part of data cleaning one should somehow get an idea of what kind of noise the measurements exhibit, how severe it is, and if possible devise some way to get rid of it. In signal processing, noise is typically assumed to be additive, for example due to the imperfect nature of the transmission channel. Such noise can be removed by averaging several measurements from the same object. In other application areas, some other kinds of aggregations may be more appropriate.

Ordered data sets may exhibit ordered disturbances in the signal content. For example level-changes (steps and ramps) which make one part of the ordered data set completely different from the other parts. Level-change may be very hard to handle because it may be masked by noise in the data, or the change may be slow. In time-sequences another problem are time delays. Each data sample in the prepared data set should correspond to one well-defined object. If there are time delays in the system, construction of such samples may be problematic since the measurements need to be collected from different moments in time.

2.3.2 New features

Often it is not possible to collect data directly of the variables that would be most interesting. However, it may be possible to capture the necessary information from some other variables or from multiple measurements of the same variable. Actually, the whole data mining process is often about finding the correct feature: the output of the final model can be thought of as nothing more than a complicated feature calculated from the raw data.

Good features capture some essential properties relevant to the problem. Often the best place to go looking for good features is domain knowledge: intuitive knowledge of the domain experts or theoretical models from the domain literature or problem documentation. Otherwise, one can just try different kinds of stuff: linear and non-linear transformations, filters, differences, etc. There are also formalized approaches to feature selection [15, 19].

2.3.3 Transformations

Transformations are necessary to convert the variables to such a form that the modeling methods can best utilize them. For example, some modeling tools can only utilize numerical information. Furthermore, many of those use Euclidean metric to measure distances between data items. For them, any symbolic variables need to be transformed into numerical in such a way that the metric makes sense.

Also some numeric variables may need to be transformed. Consider, for example, an exponentially distributed variable. The big values have huge impact on the distance metric, even if they would not be very interesting. Using a logarithmic transformation would be sensible in this case. Some modeling tools have even more strict requirements on the data, like that all values need to be between zero and one. While it is easy to scale the training data to be within this range, new data sets may hold values greater or smaller than those in the original data. These values may break the model and produce invalid results. Non-linear transformations, a sigmoid for example, may be of help in preventing this. Examples of transformation techniques include:

- linear transformations: scalings of single variables or projections of multiple variables
- non-linear transformations: histogram equalization, logarithmic, exponential or logistic scaling
- quantization/clustering
- 1-out-of-n, or m-out-of-n, coding of symbolic variables into a set of numerical variables
- mapping of symbolic variables into numerical values
- fuzzification of numerical variables into categorial or symbolic
- aggregates of multiple data instances (median, mean, sum, variance, histograms, etc.)

Many data sets are really collections or baskets of connected samples, the number of which cannot be known. For modeling purposes, the information in all connected samples needs to be collapsed into a single fixed-length sample. There are many options how to do this. The main ones are: (1) make a histogram of some sort, (2) use average or some other statistical operation for the values in each field or (3) device a sensible fixed-length concatenation of the samples. For example, sort the samples and use the concatenate of the five first as the feature vector.

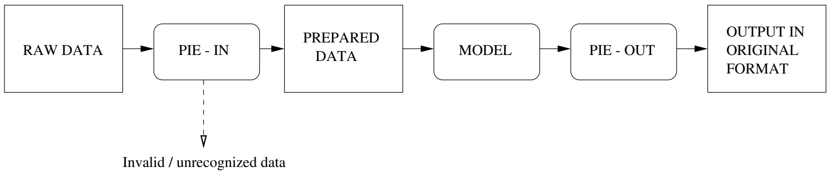


Figure 2.3: PIE in information flow.

2.3.4 Prepared information environment

The data preparation procedure must be formalized in such a manner that it can be easily repeated and, if possible, undone. Such a definition is called Prepared Information Environment (PIE) [87], see Fig. 2.3. It is important from three viewpoints:

- The iterative nature of data mining requires that the preprocessing procedure can be easily reconfigured and re-executed. To understand the results, it is essential to keep in mind what the preparation procedure has been. On the other hand, unsuccesfull preparation steps need to be undone.
- In order to apply the models to new data, the preparation procedure must be exactly the same. Otherwise the results cannot be considered valid.
- Interpretation of preprocessed variable values may be difficult. Therefore, there has to be some way to transform the prepared values into the original type and scale. Some transformations like filters are, of course, uninvertable. The output module of PIE has to define a sensible way to handle them, too.

The PIE also protects models from invalid data. All possible values are not necessarily present in the training data. Numerical variables may be out of the original value range, or a symbolic variable may present a totally new value for which no transformation exists. The input module of PIE may either report and/or discard data samples it determines as invalid.

2.4 Modeling

Modeling is the step where the solution to the problem is found. The previous steps are basically preparation for modeling and the later steps deal with its practical implementation, but the actual solution is specified at this step. This section is mostly concerned with predictive model solutions (see Eq. 2.1). However, the presented principles hold for descriptive models, as well.

2.4.1 Modeling tools

There are a number of different model families, for example linear and polynomial regression models, neural networks such as multilayer perceptron and radial basis function networks, support vector machines, fuzzy systems, classification and regression trees, etc. For further information the reader is encouraged to examine any of the numerous textbooks on the subject, for example [8, 17, 90].

The prevailing view is that there is no single best method for all problems. Each of the model families have their own strengths and weaknesses. For example, some model families have been shown to be universal approximators: with enough free parameters they can approximate any continuous function with arbitrary precision. However, this makes them also sensitive to noise. Much more robust solutions can typically be achieved by using modeling methods which have been designed to deal with the characteristics of the particular data and/or problem type.

Consider variable type, for example. If the variable is symbolic, the comparison between any two values can only state whether they are the same or different. The difference between values 1 and 2 is no different from that between 1 and 10. If, from the problem point of view, this is indeed the case, some modeling method designed to deal with symbolic variables would be the best tool, a rule base for example:

```
if a=b then predict TRUE else predict FALSE
```

Giving some meaning to the magnitude of difference would be misleading, or even harmful. On the other hand, if the magnitude of difference is important — there's a linear dependency, for example — the rule based method is clearly deficient.

In early iterations of data mining it is often advantageous to try several different kinds of models. The results give insight to the properties of the data, and later on it is beneficial if there are multiple solutions to choose from.

2.4.2 Model assessment

Different models are typically compared in terms of their accuracy. The definition of accuracy should be specified beforehand, perhaps in the problem specification step. For example, binary classification problems deal with two kinds of errors: assignment of negative instance to positive class, and assignment of positive instance to negative class. In medicine, the latter kind of error is much more serious, since extra checks are much less of a problem than a disease left untreated.

Universal approximators make very powerful modeling methods, but care should be taken as they are also prone to overtraining. When models are overtrained, they start to learn noise in addition to the real structure and, thus, loose their generalization property. Therefore, the model assessment should pay attention to the complexity of the model in addition to its accuracy [17].

Some modeling methods regularize their complexity intrinsically. They may incorporate penalty terms for high complexity in their error function, use averaging to cancel out zero-mean noise, or the model elements may be so simple that the models cannot become very complex, a linear model for example. Regularization can also be done using two sets of data. One is used for training the model, and the other is used for validating the generalization ability of the model. When the error with validation set starts to increase, the training is stopped.

To actually assess the model, a test data set is often used. This set is not used at all during the training. It is supposed to represent a "true" data set, and thus the error with test set is the "true" error of the model.

A more reliable estimation of the generalization ability of the model can be acquired by cross validation (see, e.g., [17]). There, a small set of data is set apart for validation, the model is trained using the rest of the data, and the error of the model is measured using the validation set. This is repeated several times with different validation sets, and the distribution of validation errors indicates the accuracy and generalization capability of the model. The final model is trained using the whole data.

2.5 Evaluation

Before going to the last step, deployment, the solution needs to be evaluated from the original business problem point of view. Does the solution really answer the needs of the customer? Is the solution valid with new data? Are there some important areas where the solution is especially weak? Is the solution still valid in a couple of months from now? Does it have to? There may be some evaluation criteria that are tested or which need to be met. The ultimate goal of the evalutation step is to determine whether the found solution is good enough to be deployed.

Besides the solution, the data mining process generates other results:

```
result = model + findings
```

Findings are insights, ideas, secondary models, anything that apart from the solution are important with respect to the business problem. They should also be taken a good look at.

If none of the found solutions is acceptable, it should be determined what should be done next: make better models, get more data or even redefine the problem. If the model does not seem to work well with new data, or produces strange or very bad results, one likely possibility is that the training data is biased: it lacks important information, or even has misleading information content. The findings helps in all this by hinting to still open questions, or steps which could have been done better.

2.6 Deployment

Finally, if the solution is deemed a good one, it is deployed. While actually doing this may be the responsibility of the customer, and thus out of scope of the data mining process, the data miner should provide an initial plan for doing the deployment, e.g. what kinds of software has been/needs to be implemented, how the data is gathered and processed and how the results are interpreted.

An important aspect to consider is the validity of the solution. How long and under which conditions the solution can be expected to be valid. There should be some kind of monitoring and maintainance plan enabling the customer to monitor the validity. The plan should also suggest how to proceed when the solution is no longer valid.

Also significant findings concerning the problem domain should be reported. It may have occured during the data mining process that there is valuable work to be done. Some promising options may have been left uninvestigated, or the insights gained during the process may hint to other promising data mining projects. These should be summarized in the final report.

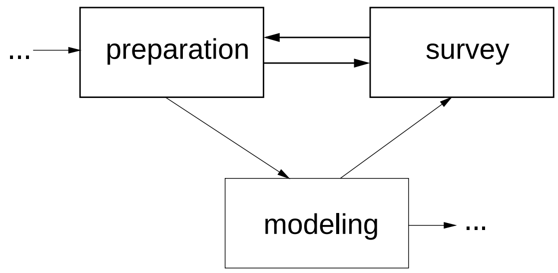


Figure 2.4: Preparation-survey –cycle.

Introduction to the CRISP-DM data mining process ends here. The next section goes back a few steps in the process. It introduces a new concept "preparation-survey –cycle" which is closely related to data understanding.

2.7 Preparation-survey –cycle

In data mining it is essential to understand the data: what are the typical values, what values can be considered unusual and where the data is unreliable? Different kinds of analysis tools can be applied but ultimately the information has to be delivered to the data miner by describing the data. The core tools in this are visualizations, summaries and (again) visualizations:

- Visualizations because they are the most efficient way to present large amounts of detailed information.
- Summaries because they compress the essentially qualitative visual information into a compact set of quantitative statements.
- Visualizations, for the second time, because the miner needs to see what kind of values underlie the summaries to judge whether they are valid [104].

2.7.1 Data survey

The purpose of data survey is to create an overview of the data: evaluate the possibilities present in the data as well as to get an overall idea, a mind model or map, of the data manifold [87]. The methods used in data survey do not aim to make a precise model of the data but rather to make sense of it.

The placement of survey in the data mining process can be seen in Fig. 2.1(c). A closer look is presented in Fig. 2.4. Data survey is closely intertwined with data preparation: the better the miner understands the data, the easier it is to prepare it appropriately. Properly preprocessed, the data is easier to understand. Of course, also modeling results can be surveyed.

Data understanding step in Section 2.2 dealt with understanding the meta-structure of the data and the reliability of (single) variables. Data survey continues from this by considering variable combinations. Examination of the data manifold involves techniques like:

- Clustering: are there natural groups in the data? Possibly each of these groups should be modelled separately.
- Dependency analysis: linear correlation, local correlation, independent components, entropy. How do the variables depend on each other? Which are the hidden factors spanning the data manifold? Are there groups of variables that are actually don't have anything to do with each other and could, thus, be handled separately?
- Statistical analysis: histograms, moments of different orders, making and testing hypothesis of the distribution. It is beneficial if the data or parts of it can be described with simple statistical models.

Visualization is an inherent part of all this. Outliers or unexpected shapes of the data manifold can easily corrupt the results of analysis if the underlying assumptions of the analysis method are not correct. Visualization is the key tool in detecting the unexpected [104].

An important consideration is also that it is not sufficient to understand a single data set well. Because the data mining process is iterative, several different data sets and preprocessing strategies need to be considered. For this reason, creating an understanding of any single data set cannot be allowed to take very long. The execution of the preparation-survey –cycle should be as smooth as possible: the data is prepared, fed into a survey module and based on the output the data can be prepared better. To make this possible, the survey tools should be as automated and therefore also as robust as possible.

2.7.2 Survey report

The report tools, visualizations and summarizations, have to facilitate the understanding task. Data characterizations have to be simple enough to be immediately understandable and the number of items that need to be conceptualized together must be as small as possible to fit into the human's working memory. Finally, the overall characterization should be as short as possible, without sacrificing accuracy too much. Examples of summary statements include:

- Distribution statements: "Variable x is within range $[a, b]$", "90% of variable x is within range $[a, b]$", "Variable x has value a" or "Distribution of x is normal $N(a, b)$".
- Property statements: "Variable x has 50% missing values".
- Dependency statements: "Variables x, y have dependency $y = kx + a$", "Variables x, y have dependency `if` $(x = a)$ `then` $y = b$" or "Variables x and y are independent".
- Cluster statements: "Cluster A has subclusters B and C", or "In cluster B $x < 1$ while in cluster C $x > 2$".

Each statement should also indicate its coverage and accuracy as well as visualize the information. For example, if distribution is being characterized, the characterization should be accompanied by a histogram plot showing the detailed information as well as indicate the meaning of the statement, for example show the estimated normal distribution curve.

In many aspects the objective of data survey methods is to estimate the probability density function of the data manifold and to convey this information to the data miner. Both tasks are very hard and both get increasingly more difficult when the dimension increases. Also the simplicity requirements imposed on the characterizations necessarily means loosing information of the details. However, data survey is more about getting rough but robust results rather than getting every detail right (which is what modeling is about). After gaining (holistic) understanding of the whole data, it is possible to return and inspect the local details more carefully.

Chapter 3

The Self-Organizing Map

The SOM has several beneficial features which make it a useful method in data mining. It implements an ordered dimensionality-reducing mapping of the training data. The map follows the probability density function of the data and is robust to missing data. It is readily explainable, simple and — perhaps most importantly — easy to visualize. Visualization of multidimensional data is indeed one of the main application areas of the SOM [58].

3.1 Basic algorithm

The basic SOM consists of M neurons located on a regular low-dimensional grid, usually 1- or 2-dimensional. Higher dimensional grids are possible, but they are not generally used since their visualization is problematic[1]. The lattice of the grid is either hexagonal or rectangular, see Fig. 3.1.

The basic SOM algorithm is iterative. Each neuron i has a d-dimensional prototype vector $\mathbf{m}_i = [m_{i1}, \ldots, m_{id}]$. At each training step, a sample data vector $\mathbf{x}$ is randomly chosen from the training set. Distances between $\mathbf{x}$ and all the prototype vectors are computed. The best-matching unit (BMU), denoted here by b, is the map unit with prototype closest to $\mathbf{x}$:

$$||\mathbf{x} - \mathbf{m}_b|| = \min_i\{||\mathbf{x} - \mathbf{m}_i||\}. \tag{3.1}$$

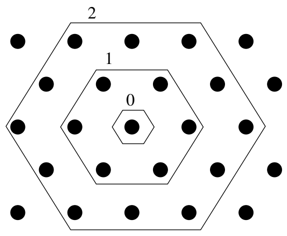
Next, the prototype vectors are updated. The BMU and its topological neighbors are moved closer to the input vector in the input space, as shown in Fig. 3.2. The update rule for the prototype vector of unit i is:

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

where t denotes time, $\alpha(t)$ is learning rate and $h_{bi}(t)$ is a neighborhood kernel centered on the winner unit. The kernel can be for example Gaussian:

$$h_{bi}(t) = e^{-\frac{\|\mathbf{r}_b - \mathbf{r}_i\|^2}{2\sigma^2(t)}}, \tag{3.3}$$

[1] There are, of course, exceptions. For example, Kiviluoto has visualized 3-dimensional map grids succesfully [55]. If visualization is not needed, even higher than 3-dimensional grids may be beneficial [102].



(a) Hexagonal grid (b) Rectangular grid

Figure 3.1: Neighborhood sets (at radius 0, 1, and 2) of the centermost unit: (a) hexagonal lattice, (b) rectangular lattice.

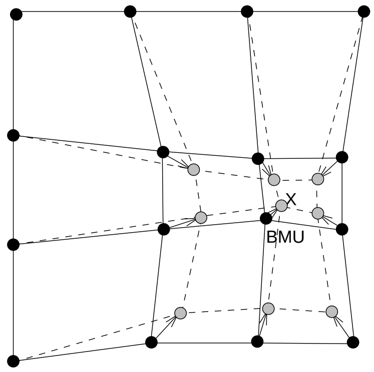


Figure 3.2: Updating the best matching unit (BMU) and its neighbors towards the input sample marked with x. The black and gray circles correspond to situation before and after updating, respectively. The lines show neighborhood relations.

where $\mathbf{r}_b$ and $\mathbf{r}_i$ are positions of neurons b and i on the SOM grid and $\sigma(t)$ is neighborhood radius. Both learning rate $\alpha(t)$ and neighborhood radius $\sigma(t)$ decrease monotonically with time. 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 prototype vectors of neighboring units resemble each other.

3.2 Variants of SOM

The basic SOM algorithm has a number of variants. The common factor is that all of them are essentially a collection of prototype vectors and a set of neighborhood relations defined between them. The prototype vectors are iteratively adjusted to correspond to the training data, and the neighborhood relations are used in such a manner that neighboring prototype vectors 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 many variants, 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 well-defined low-dimensional positions, but this location is somewhat flexible [2, 9, 31, 47, 89, 93].

Batch map. Batch map is a version of the SOM algorithm where the learning rate α is not used [58]. Also this algorithm is iterative, but instead of using a single data vector at a time, the whole data set is presented to the map before any adjustments are made — hence the name "batch". In each training step, the data set is partitioned according to the Voronoi regions of the map weight vectors, i.e. each data vector belongs to the data set of the map unit to which it is closest. After this, the new weight vectors are calculated as:

$$\mathbf{m}_i(t+1) = \frac{\sum_{j=1}^{N} h_{bi}(t)\mathbf{x}_j}{\sum_{j=1}^{N} h_{bi}(t)}, \tag{3.4}$$

where b is the BMU of data sample $\mathbf{x}_j$, see Eq. 3.1. The new weight vector is a weighted average of the data samples, where the weight of each data sample is the neighborhood function value $h_{bi}(t)$ at its BMU b.

Tree-structured SOM. Tree-structured SOM is an especially fast version of the SOM [64, 63]. It consists of a set of layers, each of which is a complete quantization of the data space. The difference between layers is that the number of prototypes grows exponentially as the tree is traversed downwards. For example, the first layer has only 4 prototype vectors, the second has 16, the third 64, and so forth. Thus, each prototype vector of a layer has four descendants in the next layer. The top layers are utilized in the training of latter ones: instead of comparing a given data vector to all prototype vectors of, say, layer 3, it is first compared with prototypes in layer 1, then with the descendants of the 1st layer winner and its neighbors, and so on. The number of distance calculations is significantly reduced especially on the lower layers. Also, layers are added one at a time so one gets a gradually more detailed mapping of the data.

MST-SOM. In MST-SOM the neighborhood relations are defined using a minimal spanning tree (MST) [48]. Minimal spanning tree defines the shortest possible set of connections which link together a set of vectors. In vector quantization, MST-SOM is much faster and more stable than the basic SOM. On the other hand, the prototypes no longer have well-defined positions on a low-dimensional grid and thus visualization is more problematic.

Neural gas. Neural gas is another variant of the SOM where the neighborhoods are adapatively defined during training [77]. Neighborhoods are defined by the ranking order of the distance of prototype vectors from the given training sample.

Growing Cell Structures. In growing cell structures algorithm the adaptability has been taken one step further [29, 30]. Instead of having a fixed number of prototype vectors, the algorithm starts with only two, and then adds new prototype vectors according to an error function criterion. The neighborhoods are defined at the time a new prototype is added into the network. Prototype vectors can also be removed.

3.3 Related algorithms

Kernel methods. In general, the SOM is related to kernel methods which approach data modeling using prototypes [17]. Examples of kernel methods used for modeling include radial basis function networks [86, 8] and ellipsoidal fuzzy systems [66].

Possibly the closest well-known relative of the SOM is the k-means vector quantization and clustering algorithm. Note that the difference between classical vector quantization [34, 81] and SOM is that SOM performs local smoothing in the neighborhood of each map unit, as can be seen from the distortion measure in Eq. 3.8. If the neighborhood kernel value is one for the BMU and zero elsewhere ($h_{bi} = \delta(b, i)$ in Eq. 3.2), the SOM reduces to adaptive k-means algorithm [81].

Another closely related algorithm is the fuzzy c-means algorithm [5], 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. Also the SOM can be interpreted in this way if the neighborhood function values are used as membership values.

Principal curves. Principal curves and surfaces [37] represent a very similar concept to the SOM [17, 49]. In principal curves, the idea is to find the central curve (or surface) going through the data manifold. Each point on the principal curve is the average of all points that project to it. The SOM prototype vectors can be interpreted as conditional averages of the data, and thus it is a discrete counterpart of the principal curve [91].

Vector projection. The SOM is also related to vector projection algorithms [49, 17]. In vector projection one tries to find low-dimensional coordinates for high-dimensional data samples such that certain features of the original data set are preserved as well as possible. The goal is dimensionality reduction, and often visualization if the output space is 2- or 3-dimensional. Typically, the features to be preserved are pairwise distances between data samples (or at least their order) and subsequently preservation of the shape of the data manifold in the projection. In multi-dimensional scaling (MDS) [68] all pairwise distances are weighted equally. The energy function to be minimized is:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij} - d'_{ij})^2, \tag{3.5}$$

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. The bigger distances have larger effect on the error function, and thus global organization is much more important than local topology. In

Sammon's mapping [95] and Curvilinear Component Analysis (CCA) [21] the nearby samples are weighted more, and thus local topology is preserved better:

$$\text{Sammon's mapping:} \quad E = \sum_{i=1}^{N}\sum_{j=1}^{N}(d_{ij} - d'_{ij})^2/d_{ij} \tag{3.6}$$

$$\text{CCA:} \quad E = \sum_{i=1}^{N}\sum_{j=1}^{N}(d_{ij} - d'_{ij})^2 e^{-d'_{ij}} \tag{3.7}$$

Note that Sammon's mapping emphasizes closeness in the input space, whereas CCA emphasizes output space. Other vector projection algorithms can be found from [23, 53, 65, 76, 97]. The SOM is also a vector projection algorithm because the prototype vectors have well-defined positions on the low-dimensional map grid.

Energy function. When comparing the SOM with other algorithms, the energy function of the SOM — or the lack of it — inevitably comes up. It has been shown that the basic SOM algorithm has no energy function in the general case [26]. In case of a discrete data set and fixed neighborhood kernel, the map distortion measure

$$E = \sum_{j=1}^{N}\sum_{i=1}^{M} h_{bi}||\mathbf{x}_j - \mathbf{m}_i||^2. \tag{3.8}$$

can be shown to be a local energy function of the SOM [57]. However, when the BMU index b of any of the data samples $\mathbf{x}_j$ changes, the energy function changes slightly, and thus the SOM only gives an approximate solution to Eq. 3.8. To obtain the exact solution, the definition of winner (Eq. 3.1) should be changed to $b = \arg\min_k\{\sum_i h_{ki}||\mathbf{x}_j - \mathbf{m}_i||^2\}$ [39]. This is computationally much heavier than the basic SOM, but it offers an interesting interpretation of the map as a way to construct noise-resistant coding of the input vectors. If the neighborhood function is normalized $\sum_k h_{ki} = 1$, the values give the noise-induced transition probabilities between codes (map units) [74, 33].

3.4 Data analysis using SOM

3.4.1 Quantization

The SOM has properties of both vector quantization and vector projection algorithms. The quantization from the N training samples to M prototypes reduces the original data set to a smaller, but still representative, set to work with. Further analysis — for example clustering or visualization, see Fig. 3.3 — is performed (primarily, or at least initially) using the prototype vectors instead of all of the data.

Using the reduced data set is only valid if it really is representative of the original data. For classical vector quantization it has been shown that the density of the prototype vectors is proportional to $c \cdot p(\mathbf{x})^{\frac{d}{d+r}}$, where $p(\mathbf{x})$ is the probability density function of the input data, d is dimension, r is distance norm and c is normalization constant [32, 59, 118]. For SOM, a similar power law has been derived in one-dimensional case [92]. Even though the power law holds only

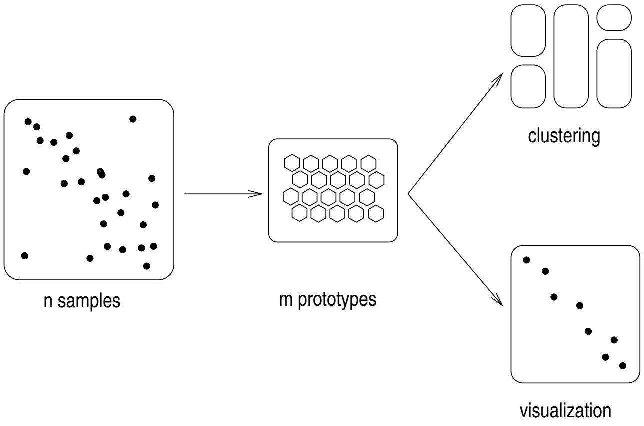


Figure 3.3: Data analysis using SOM as an intermediate step.

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 [59]. 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 roughly follows the density of the training data.

This approach is closely related to data squashing where the idea is to scale data sets down rather than make new algorithms which scale up to the data sets. Data squashing is based on keeping the (local) statistics of the data set as close to the original as possible [25]. In vector quantization the probability density function guides the process, which amounts to much the same thing.

The primary benefit of using a reduced data set is that the computational complexity of subsequent steps, for example clustering, is reduced [112]. The reduction is important especially in early steps/iterations of data mining. Because several different data sets and preprocessing strategies need to be considered speed and robustness are more important than accuracy. In building final models accuracy is the most important consideration, and the reduced data set may be discarded in favour of a bigger one, or even the whole data.

Another benefit of vector quantization is that it usually involves averaging of data samples, thus removing zero-mean noise and reducing the effect of outliers.

3.4.2 Projection

To be able to visualize the prototypes efficiently, vector projection is needed. Together the set of prototype vectors and their projections form a low-dimensional map of the data manifold.

Since the prototype vectors of the SOM have well-defined positions on the low-dimensional map grid, the SOM is a kind of vector projection algorithm. The projection of a data sample can be defined to be the index b or location $\mathbf{r}_b$ of its BMU on the map grid. This kind of projection is of course very crude. The projection is discrete as it can only get as many values as there are map units. Therefore, different vectors may be projected to the same point. Also, since the shape of the map is defined beforehand, information of the global shape of the data manifold is lost. In contrast, most vector projection techniques, like Sammon's mapping [95], have continuous outputs. Thus, in certain visualization tasks the projection defined by SOM is often complemented with other methods, see Section 4.2.2.

However, as a projection algorithm, SOM has an important advantage over many others. The topological ordering of map units depends primarily on the local neighborhood which is defined on the map grid. Since there are more map units where data density is high, the neighborhood in these areas becomes smaller as measured in the input space. Thus, the projection tunes to local data density.

3.4.3 Benefits and pitfalls

There are also other algorithms which combine properties from vector quantization and vector projection, for example Vector Quantization and Projection neural network [22], and combinations of any vector quantization and vector projection algorithms, like k-means and Sammon's mapping [28] or k-means and MDS [96]. The data mining methodology explained in this thesis can be applied using any of these other techniques as well. Which of them is the best one depends entirely on the intended application, because each algorithm has a different goal.

The goal of SOM is to create a topologically (i.e. locally) ordered mapping of the data in the sense of a discretized principal surface or curve. In addition to the computational benefits offered by vector quantization, the primary benefits of SOM are:

- *Robustness.* Assuming the neighborhood function reaches far enough, like in the case of Gaussian neighborhood function, the SOM is very robust since all prototypes are affected by all data samples.
- *Local tuning.* The topological ordering works primarily in the neighborhood of each map unit, and therefore tunes locally to the data density.
- *Ease of visualization.* The regular map grid makes it easy to build efficient visualizations and user interfaces.

Many of the above benefits are due to the neighborhood relations. They are essential in order to create the organization on SOM. However, the neighborhood also has three unfortunate side-effects.

- *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.
- *Contraction.* The range of variable values is contracted. The averaging made by the vector quantization procedure, and enhanced by the neighborhood function, averages the extreme values out. This may be an undesirable side-effect in some situations, for example if outliers are interesting from the analysis point of view.

- *Interpolating units.* When the data cloud is discontinuous, interpolating units are positioned between data clusters providing convenient extrapolative estimates of 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 de-emphasized or completely left out [112].

3.4.4 Scalability

Fig. 3.4 shows the implementation of one SOM training step as C-code. The computational complexity of one training step is $\mathcal{O}(Md)$. Correspondingly, one epoch of training — going through the data once — has complexity of $\mathcal{O}(NMd)$, where N is the number of data samples. The complexity of the whole training process depends on the number of training epochs. If this is chosen to be proportional to M/N (or equally, the number of training steps is proportional to M as in [60]), the complexity of the whole training is $\mathcal{O}(M^2d)$.

Furthermore, if the number of map units is chosen to be proportional to $\sqrt{N}$ as in [112], the complexity of the whole training scales linearly with the number of data samples $\mathcal{O}(Nd)$. 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, as in [60].

The memory consumption depends on whether the interunit distances in the output space are calculated beforehand or not. If they are, the memory consumption scales quadratively with the number of map units. If not, the consumption scales linearly, but $6M$ additional floating point operations per training step are needed (assuming a 2-dimensional map grid is used). Note that the memory requirements due to the training data have been ignored. While just one training sample is needed at a time, in practice as much of the data as possible is usually kept in the main memory to reduce the overhead due to disc-access (or some other mass storage device) time.

Thus, the SOM is applicable also to relatively large data sets. Training huge maps is time-consuming, 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 or by searching for the winner only in a low-dimensional subspace of the input space [50, 60, 63]. Note also that the training algorithm can be easily implemented both in a neural, on-line learning manner as well as parallelized [73, 61].

```
/*** find the BMU ****************************************************/
bmu=0; min_di=1000000; /* or some other big enough number */
for (i=0; i<M; i++) {  /* M = number of map units, d = dimension */
  for (di=k=0; k<d; k++) { f = x[k] - m[i][k]; di += f*f; }
  if (di<min_di) { min_di=di; bmu=i; }
}
/*** update ***********************************************************/
for (i=0; i<M; i++) {
  h = alpha*exp(delta(bmu,i)/radius); /* Gaussian neighborhood */
  for (k=0; k<d; k++) m[i][k] -= h*(m[i][k] - x[k]);
}
```

Figure 3.4: One step of SOM algorithm as C-code. In the code, `x[k]` is the kth component of data sample $\mathbf{x}$ and `m[i][k]` is the kth component of prototype vector $\mathbf{m}_i$. The `delta` is a table of squared map grid distances between map units $\|\mathbf{r}_b - \mathbf{r}_i\|^2$, `radius` equals $-2\sigma(t)^2$ in Eq. 3.3 and `alpha` is the learning rate $\alpha(t)$ in Eq. 3.2. Thus, ignoring calculation of `radius` and `alpha`, and assuming that the map grid distances `delta` have been calculated beforehand, there are $(6d+2)M$ floating point operations (additions, substractions, multiplications, divisions or exponents). The memory consumption is Md floating point numbers for the prototype vectors and M^2 for the `delta` matrix.

Chapter 4

Using SOM in data mining

The use of the SOM in exploratory data analysis has been studied previously. Kaski presents a compact overview of the properties of the SOM from the point of view of data exploration and compares the SOM to various related algorithms [49]. In [20] a number of data analysis cases is presented where the SOM has been an important tool. More examples of fruitful usage of the SOM in various engineering tasks can be found for example from [62, 98]. Important from data mining point of view is postprocessing of the SOM such that the data engineer can efficiently interpretate the results, done for example in [51, 71, 83, 84, 88, 105].

Fig. 4.1 shows how SOM fits in the preparation-survey –cycle. Vector quantization divides the data space to a set of representative prototypes. Using automated clustering methods and the qualitative information given by the visualizations one can select subareas and subspaces from the data and perform quantitative analysis on these rather than blindly on the whole data. Some areas may be discarded as outliers, some variables may be left out or preprocessed in a more appropriate manner. And then the data can be reanalysed.

If the preparation seems adequate, it is easy to build initial models — local models or non-parametric nearest neighbor models — using the prototypes or a set of data selected based on the prototypes. In general, nearest neighbor methods are known to be robust and to produce relatively good models [45].

All in all, this approach offers a convenient workbench where to create an initial understanding of the data at hand, as well as to produce some initial models.

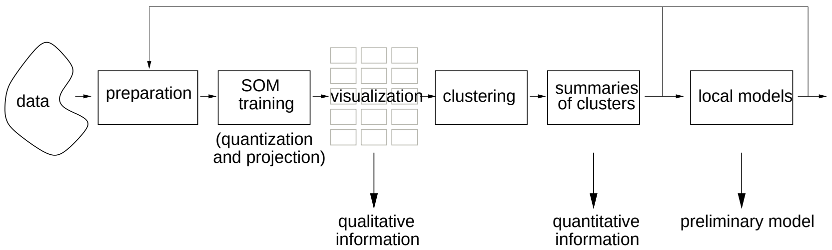


Figure 4.1: Using SOM in preparation-survey –cycle.

In the rest of this chapter, various ways to utilize the SOM in different steps of the data mining process are discussed. Basically, the SOM is a clustering and projection tool so these stand out among the applications. These properties can be utilized for example in various transformations. Combined with local models the SOM can be used for modeling and probability density estimation. The SOM has also been used as a classification tool, although for this purpose it is obviously suboptimal since it doesn't make efficient use of class information to position the prototypes near class boundaries (unlike, for example, LVQ [58]). The main focus in the following is, however, in how the SOM can be best utilized in the preparation-survey -cycle. As such, special attention is given to visualization, clustering and summarization.

4.1 Preparation

Data reduction. Naturally, the SOM can be used for dimensionality reduction, but it can also be used for data selection. Instead of using all data vectors of the original data set, one could use only a certain percentage of the samples in the Voronoi set of each map unit. If certain kinds of samples need to be sampled more than the others, the areas of the map corresponding to these samples may be sampled with a higher percentage. Of course, this sampling can be done only after the SOM has been trained.

Discretization. Because of the discrete output set, the SOM suits very naturally to binning and fuzzification. Because more prototype vectors are positioned to the areas where there is a lot of data, the binning can be thought of as multi-dimensional histogram equalization.

Symbolical to numerical. The SOM can also be used for mapping symbolical variables into numerical values. While the SOM algorithm cannot utilize symbolical variables, their positions on the map can be analyzed afterwards. This is done by finding the BMU of each data vector using the existing numerical variables, "labeling" the BMU with the symbolical values and finally analyzing the distribution of the values on the map grid. If different symbolical values have distinct places on the map grid, the map grid coordinates can be used to replace symbolical values with numerical counterparts.

Aggregation. Another use for the SOM is as an aggregation method. Multiple data vectors can be projected on a SOM and the resulting hit histogram or response pattern can be used as an aggregate value for the projected data set (see Section 4.2.2). In case of complicated data sets, many such SOMs can be combined to form hierarchical maps [116]. The difficulity with this approach is interpretation, and therefore it is beneficial if the number of bins in the histogram is as small as possible. This implies the SOM may need to be clustered before creating the aggregates.

Missing value replacement. The SOM training algorithm can handle missing values gracefully. In searching the BMU, only the values which have known values are used in the distance calculations. Note that this implies that the missing values are assumed to be (almost) the same as the values present in the prototype vectors [94].

When using the SOM for replacing missing values, the replacements can be taken directly from the prototype vectors of the BMUs. This corresponds to using conditional estimates as missing value replacements, where the estimate is based on the known values. If a more advanced system is desired, each map unit can have a simple local model which can be used to calculate the missing values.

4.2 Understanding data

The SOM forms a low-dimensional map of the training data. The ordered SOM grid can be used as a convenient visualization platform for showing different features of the SOM (and thus of the data).

However, often the qualitative information provided by the visualizations is not enough. To produce quantitative summaries of the properties of the data, interesting groups of map units can be selected from the SOM. One such interesting group is of course the whole map. If it seems that there are two or more qualitatively different regions on the map, they should be analyzed separately. In effect, the SOM needs to be clustered (or segmented) either by hand but preferably automatically. After clustering, the properties of the clusters can be summarized.

4.2.1 Visualization

The goal of visualization is to present large amounts of detailed information in order to give a qualitative idea of the properties of the data [104]. In general, efficient visualization is not only extremely domain-dependent but also culture-dependent. The special properties and historical conventions of each application dictate what kind of visualizations transmit the intended information most efficiently and reliably. In this thesis the aim is to visualize generic vector-valued data objects. The actual visualizations consist of a number of these objects — the prototypes and possibly data samples — and the objective is to give an idea of their properties and relations.

One of the problems in visualization of multidimensional information is that the number of visual dimensions is limited. Visual dimensions are those properties of the visualization that can be controlled to show different values[1]. To

[1] Chernoff's faces are a peculiar example of a set of visual dimensions: data samples are shown as faces such that the values of the variables control different features of the face: lengths, positions, curvatures and angles of nose, eyes, mouth, ears and chin [18]. While this is an intriguing idea, it is not very useful in practice because of several reasons.

1. 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.
2. The features do not map arbitrary values very well, say how does date of month translate into angle of nose?
3. There's a hierarchy in the importance of features which is hard to quantify. The human observer pays much more attention to eyes than, say, ears.
4. The faces are not particularly easy to produce using standard software.

be efficient, the distinguishing features in the visual dimensions should be such that the human visual system detects them fast and with little effort, i.e. in the preattentive processing phase [38]. Examples of such features include position, form (size, shape and orientation) and color (hue, texture, intensity).

Typically the number of properties that need to be visualized is much higher than the number of usable visual dimensions. It is simply impossible to show them all in a single figure or visualization. The solution is to use a number of visualizations that have been linked together so that one can immediately identify the same object from the different visualizations [14]. When several visualizations are linked in the same manner, scanning through them is very efficient because they are interpreted in a similar way. This kind of visualizations are called small multiples [104].

Objects in small multiples can be linked together using for example similar position: each object is in the same position within each multiple as in Fig. 4.2(a). Other visual dimensions, for example color or shape, are then used to show the properties of the objects. Also color can be used for linking: the same object has the same color in each of the visualizations, but the position changes as in Fig. 4.2(b). Both color and similar position have, however, limited accuracy: it requires considerable effort to determine whether two points in even close-by multiples are in exactly the same position, or have exactly the same color. To achieve greater accuracy in linking one can use identical position: visualizations are positioned on top of, or inside, each other. Obviously this technique can only be used only for a limited number of visualizations since they essentially occupy the same space.

4.2.2 Visualization of SOM

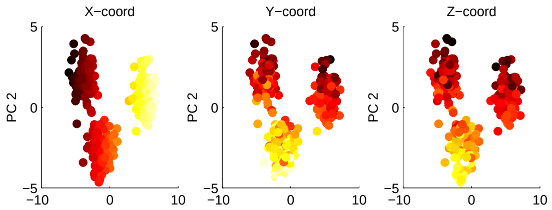
There are a number of techniques for visualization of SOM. Based on their goal, they can be divided to three groups: visualization of cluster structure and shape, visualization of components and visualization of data on the map.

Cluster structure and shape

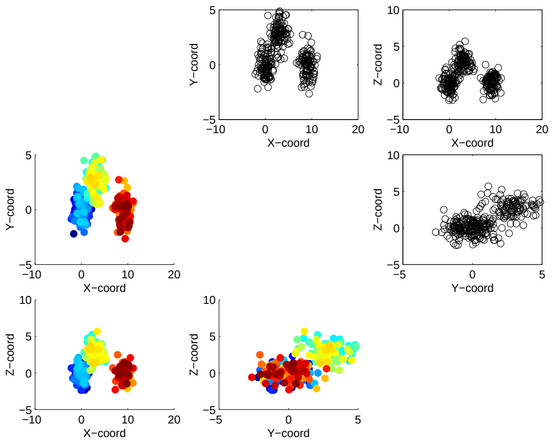
Visualization usually starts with trying to give an overall idea of shape of the map in the input space. Especially, are there clusters and if so, how are they related to each other? Fig. 4.3 shows four different ways of visualizing the cluster structure of the map.

Distance matrices. The most commonly used visualization technique to detect clusters from SOMs are distance matrices: a matrix of distances between neighboring map units. The distance matrix can either hold all distances between map units and their immediate neighbors, like in the U-matrix [106], or just a single value for each map unit, for example the median of the distances to the neighbors [67], see Fig. 4.3(a) and (b). Because neighboring map units are typically also very close to each other in the input space, distance matrices are related to single linkage clustering. The distance matrices can be visualized using for example gray shade, but also other techniques are possible [35, 44, 79].

Distance matrices essentially show the density of prototype vectors in different parts of the map. A recently proposed method uses the distribution of data



(a) Similar position



(b) Color

Figure 4.2: Linking small multiples. In (a), a 3-dimensional data set of three Gaussian spheres is presented in three small multiples. The multiples are linked by similar position, the coordinates of each sample having been defined by a projection to the space spanned by the two greatest eigenvectors of the data set. Colors show the corresponding variable values. In (b), there are six small multiples which show the scatter plots of each variable vs. another. The ones on bottom left are linked using color.

samples around the prototype vectors to enhance the distance matrix technique [83]. The distance matrices can be enhanced in other ways as well, for example by defining the distances in ways other than the Euclidian distance, for example along the minimum spanning tree.

Colormaps. A related technique is to give similar colors to similar map units [41, 52, 53], see Fig. 4.3(c). The method introduced in [53] even takes perceptual differences of the colors into account in order to reflect the distances in the original data space as faithfully as possible. These colormaps are useful, not only as a clustering visualization, but also because they can be used to create color codings which take the cluster structure into account. The color coding can then be used for linking different visualizations together [40, 52].

Projections. A different approach is to use projections of the prototype vectors. This makes it possible to visualize not only the cluster structure but also the overall shape of the data manifold. Note that the projection defined by the SOM is too rigid to be useful for this purpose. Some more flexible method, Sammon's mapping [95] for example, must be used instead.

The position information can be complemented by connecting objects that are, for example, neighbors. Particularly useful this is in 3D if proper depth-cues cannot be shown: lines connecting the objects can give the visualization a shape that single objects hanging in space fail to convey. Visualizing the SOM by projecting the prototype vectors to a low dimension and connecting each unit to its neighbors gives the map its characteristic net-like look and makes it easy to grasp its shape, see Fig. 4.3(d).

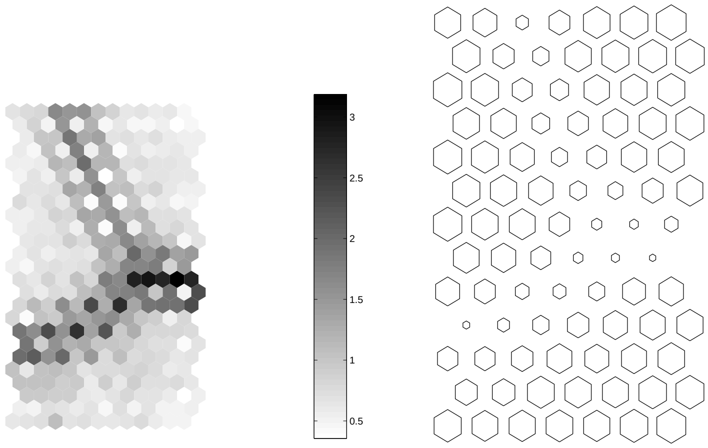
Components of the map

After getting an overall idea of the shape of the data manifold, it is possible to concentrate on smaller details:

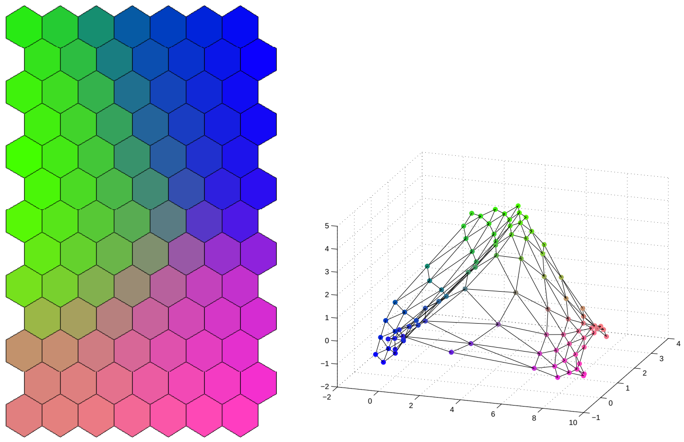
- What kinds of values do the variables have?
- What kinds of values or value combinations are typical for different parts (clusters) of the data manifold?
- Are there significant dependencies between the variables?

The basic technique in this is the visualization of component planes, see Fig. 4.4. Each component plane can be thought of as a slice of the map: it consists of the values of a single vector component in all map units. Each component plane visualizes the spread of values of that component. As such they are like histograms, the difference being that the same value can be present in multiple places of the map if it is typical to several different clusters. Coupled with the clustering information, the component planes show the values of the variables in each cluster.

It is easy to detect possible correlations between variables. By comparing component planes with each other, correlations are revealed as similar patterns in identical positions of the component planes: whenever the values of one variable change, the other variable changes, too [111]. Based on overall view, it is easy to select interesting component combinations for further investigation. A



(a) U-matrix (b) Distance matrix



(c) Similarity coloring (d) Projection

Figure 4.3: Clustering visualizations: (a) U-matrix with values indicated using grayscale, (b) distance matrix using hexagon size to show the values, (c) similarity coloring, (d) the map network in 3-dimensional space. The colors in (c) and (d) match each other.

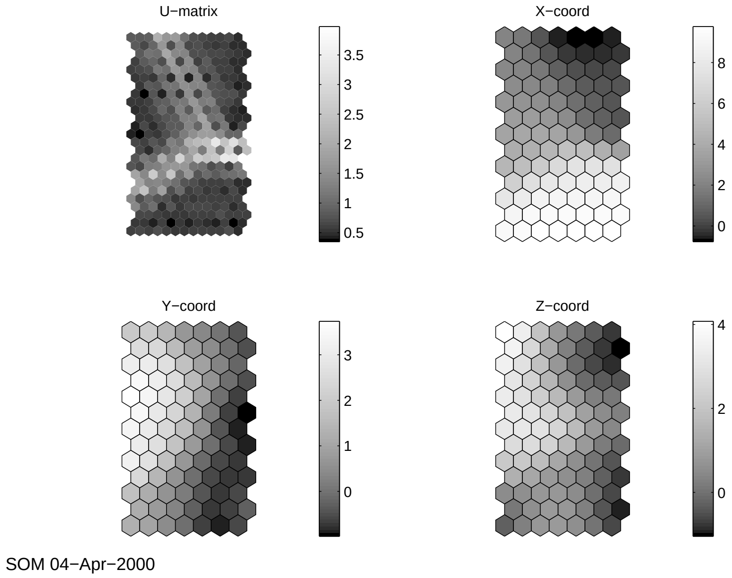


Figure 4.4: U-matrix (top left) and three component planes. The different plots are linked together using similar position. It can be seen, for example, that high "X-coord" values are typical for the cluster on the bottom of the map.

more detailed study can be done using e.g. scatter plots [40]. Note, though, that correlation hunting is not what these techniques have been desinged for. The perceived (partial) correlations may actually be artifacts induced for example by the cluster structure of the map.

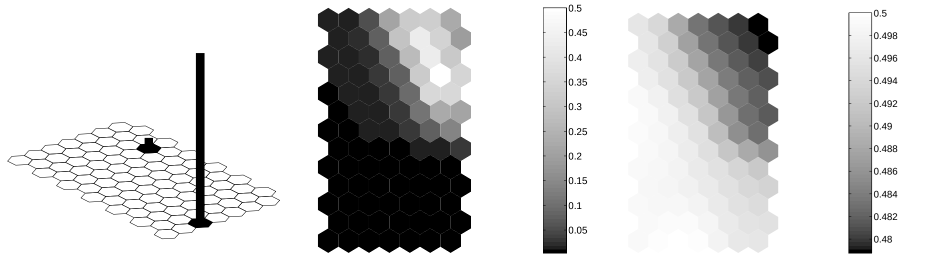
A very interesting property of a cluster is, of course, what makes it a cluster. Methods for visualizing the contribution of the original variables to the cluster structure have been developed by Kaski *et al.* [51, 83].

Data on the map

Investigation of the relationship between the SOM and data vectors is interesting because it offers a way to identify regions from the map based on familiar data samples, and to check how well the mapping has succeeded. Besides training data, the investigation of new data samples is also important. The objective may be, for example, classification or novelty detection.

Fundamental to these techniques is the definition of the location of a given data sample on the map, or the response of the map to the data sample. Typically, the location is the BMU of the sample, as defined in Section 3.4.2. Alternatively, the response can be calculated based on a generative mixture model calculated for the map [3]. For multiple vectors, a data histogram is obtained. In case of time-series data, the adjacent BMUs can be combined to form a trajectory on the map [54, 103].

However, the BMU is not the whole truth, since simply indicating it from the map completely ignores the accuracy of the match. Typically there are several prototypes which match the data sample almost as well as the BMU. If these



(a) (b) (c)

Figure 4.5: Projection visualizations. In (a) the BMUs of two samples are indicated from the map with hexagons. The vertical bars show the match accuracies. The height is proportional to the ratio between quantization errors $q_i = ||\mathbf{x} - \mathbf{m}_i||$ to best- and worst-matching prototype vectors $\min_i q_i / \max_i q_i$. For the one on front this ratio is $165/173 = 0.96$, and for the one on back it is $0.59/9.8 = 0.06$. In (b) and (c) are fuzzy responses of the two samples calculated as $1/(1 + (q_i/a)^2)$, where a is a normalization constant. Notice that in (b) the colorscale values are between [0,0.5] while in (c) they are between [0.48,0.5].

map units are at different parts of the map this either means that there's a fold on the map, or that the data sample is far away from the data manifold modelled by the map.

Fig. 4.5 shows two ways to visualize the response of the map to two data samples, other than simply pointing out the BMU. Either the properties of the BMU marker are used to indicate the accuracy, or the response of each map unit is shown [110, 115]. The latter gives more information, but the former is much more compact and allows visualization of multiple responses in the same figure.

Notice that by comparing the responses data samples, or even whole data sets, can be compared to each other within the context provided by the map [116].

4.2.3 Clusters and summaries

In order to produce quantitative information, interesting groups of map units (or data samples) can be summarized. Summary of the whole data set is, of course, essential, but if the data manifold actually consists of two or more separate clusters, the properties of these clusters and how they differ from each other is very interesting. Based on the SOM, (the Voronoi set of) each map unit could be summarized separately. However, this would produce way too many summaries to be useful. Therefore, the map needs to be segmented into a smaller set of "natural" clusters — small enough to keep the conceptualization of the result as easy as possible.

Clustering for summarization

Clustering is about finding "interesting" groups from the data [17]. In crisp clustering, each data sample belongs to exactly one cluster. Fuzzy clustering [7] is a generalization of crisp clustering where each sample has a varying degree of membership in all clusters. If desired, the fuzzy clustering can be defuzzified by assigning each sample crisply to the cluster where its membership (or probability, if mixture models are used [78]) is highest.

A widely adopted definition of optimal clustering is a partitioning that minimizes distances within and maximizes distances between clusters. However, this leaves much room for variation, since the within- and between-clusters distances can be defined in several ways [6]. In practice most clustering methods do not produce a single clustering, but offer several with different number of clusters. To select the best among them, each can be evaluated using some kind of validity index. As can be expected there is also a multitude of different validity indeces [80, 6].

It should be emphasized that the goal here is not to find an optimal clustering for the data — in some specific sense — but to get a robust insight into the structure of the data for the purpose of data survey. Actually, the clusters need not even form a proper partitioning: some points can well be discarded as "unclustered" and the groups may form a hierarchy with super- and subclusters. Since the purpose is to gain insight into the data, it is sufficient if the found groups are something that the data engineer can comfortably identify as a separate or at least homogeneous part of the data cloud [112].

Clustering algorithms

Because the definition of optimal clustering is far from fixed, there are a huge number of different kinds of clustering algorithms. However, the two main ways to cluster data are hierarchical and partitive approaches [46]. The hierarchical methods can be further divided to agglomerative and divisive algorithms, corresponding to bottom-up and top-down strategies.

Agglomerative clustering algorithms typically merge data points together one at a time to form a clustering tree which finally consists of a single cluster, the whole data set. The clustering tree, dendrogram, can be utilized in interpretation of the data structure and determination of the number of clusters, see Fig. 4.6. However, the dendrogram does not provide a unique clustering. Rather a partitioning can be achieved by cutting the dendrogram at certain level(s), for example using some kind of interactive tool [11]. Divisive algorithms are similar but work in the opposite direction starting from a single cluster and dividing each cluster to subclusters until all vectors are in a cluster of their own.

Partitive clustering algorithms divide a data set into a 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 the energy function [13]. If the number of clusters is unknown, the partitive algorithm can be repeated for a set of different values, typically from 2 to $\sqrt{N}$, where N is the number of samples in the data set. An example of a commonly used partitive algorithm is the k-means.

Partitive methods are less sensitive to imperfections — noise, outliers and even overlapping clusters — in the data than hierarchical methods [75]. On the

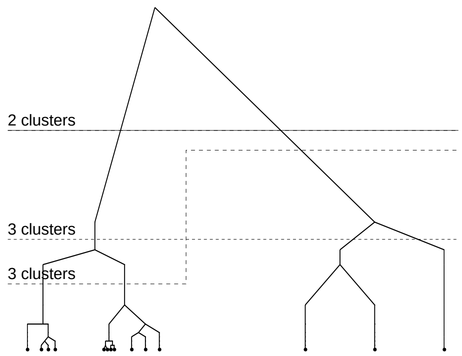


Figure 4.6: A dendrogram of a set of 14 one-dimensional data samples. The samples are on the bottom of the figure, positioned according to their value. Initially each is in a cluster of its own. One by one, the clusters are joined together, as indicated by the lines. The height indicates the distance between the two clusters when they were joined. A partitioning can be obtained by cutting the dendrogram at a certain level, for example at the level where there are only two clusters left. An alternative way is to cut the dendrogram at different level for each branch.

other hand, partitive methods make implicit assumptions on the form of the clusters. For example, k-means tries to find spherical clusters.

SOM as a clustering algorithm

SOM is also a clustering algorithm: each map unit defines a small protocluster consisting of the samples in its Voronoi set. Alternatively, the SOM can be interpreted as a fuzzy clustering: each data sample is part of every cluster with membership value proportional to the value of the neighborhood function at its BMU. This interpretation can be useful if the number of samples per protocluster is small or if a fuzzy method is used as a postprocessing stage based on the output of SOM.

However, unlike most other prototype-based methods, the optimal situation for SOM is not when the number of prototypes equals the number of clusters. Instead, the number of units in the SOM should be much bigger than the expected number of clusters. The neighborhood function draws neighboring map units together, and thus neighboring map units reflect the properties of the same rather than different clusters. The transition from one cluster to another on the map is gradual taking place over several map units. This means that if a small amount of clusters is desired, the SOM units need to be clustered [72, 112].

Using SOM (or some other vector quantization algorithm) as an intermediate step makes the clustering a two-level approach: the data set is first clustered using the SOM, and then the SOM is clustered. Each data vector of the original data set belongs to the same cluster as its nearest prototype. One big benefit of this approach is that computational load decreases. Even with relatively small number of samples many clustering algorithms — especially hierarchical ones

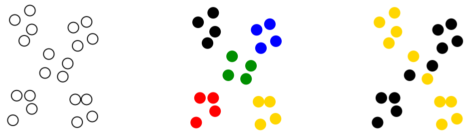


Figure 4.7: The original data on the left, a typical clustering result in the middle and an alternate clustering result on the right. Assuming the data actually consists of two crossing lines, the correct result is the one on the right: in one cluster, there is a very clear positive correlation, and in the other equally clear negative correlation. However, the more likely clustering result in the middle has five clusters none of which shows any correlation between variables. Also, summarization of the whole data shows no correlation, since the two opposite ones cancel each other out.

— become intractably heavy. For this reason, it is convenient to cluster a set of prototypes rather than directly the data [107].

Either partitive or hierarchical methods can be applied to cluster the SOM. The prototypes may either be clustered directly, or some specific features of the SOM can be utilized in the clustering. For example, the interpolative units (see Section 3.4.3) indicate gaps, i.e. cluster borders, in the data manifold [119]. In partitive clustering the interpolating units can be left out of the analysis. In agglomerative clustering the SOM neighborhood relation can be used to constrain the possible merges in the construction of the dendrogram [82]. If this is used together with the neighborhood constraint, the interpolative units form borders on the map which the construction of the dendrogram must obey. It may even be that the interpolating units completely separate some areas from the rest of the map.

One can also use the distance matrix directly as a basis for clustering. Since the distance matrix, for example average distance of each prototype vector to its neighbors, gives a rough estimate of the local probability density, the local minima of the matrix can be used as the centers or seed points of the clusters. The partitioning can then be done either by assinging each map unit to closest center or by using region-growing starting from each local minima.

Summaries

The purpose of summarization is to give a concise characterization of the properties of the target data set, as described in Section 2.7. The approach below is hierarchical and based on clustering. Note that the clustering result has major impact on the summarization result. Consider a data set in the shape of an `'X'` depicted in Fig. 4.7. It is important to realize that most clustering methods fail in handling these kinds of overlaps. Supporting the summarization with visualizations is important to detect this kind of anomalies.

The summary starts with the desciption of the data set as a whole: distributions of and interactions between variables. In addition, the major subclusters of the data set are listed. After this, each subcluster is summarized in a similar manner. Likewise their subclusters. This hierarchical approach allows one to concentrate only on a few objects at a time, as opposed to dividing the whole data set directly to dozens of clusters.

In addition to the properties of single variables and variable pairs listed in Section 2.7, characterization of clusters can list some of the following properties.

- Typical cases (prototypes), and possibly extreme cases (archtypes).
- The shape of the cluster, especially whether it is spherical or not. If a simple statistical model (e.g. a Gaussian sphere) can be used to characterize the cluster, so much the better.
- Are there, or might there be subclusters within this cluster?

To reduce the length of the description, listing properties which are the same as with the supercluster should be avoided.

In order to be able to place the cluster in the right context, one should also relate it to the rest of the data set.

- Which are the neighboring clusters?
- What are the distinguishing properties of the cluster from the neighboring clusters (border cases)? Is there really a gap between the clusters?
- Which factors make the cluster different from the rest of the data?

4.3 Modeling

Training predictive models (see Eq. 2.1) with data is a supervised learning problem. There is an output y which needs to be constructed from the inputs $\mathbf{x}$. As opposed to this, the SOM algorithm is designed for unsupervised learning: none of the variables is given the special status that output variable y has in supervised learning. However, this is not to say that the SOM cannot be used for modeling.

The basic SOM can be easily used as a lookup-model. Taking any set of known values (the input variables), their BMU — or some other type of response — from the map can be determined, and the corresponding prototype vector can be used to give values to the other (output) variables [99]. If more than one unit is chosen, a weighted average of the values can be used. Just as in handling missing values in Section 4.1, this corresponds to prediction using conditional estimates.

Typically though, when using SOM for modeling, the input and output variables are handled separately. The unsupervised quantization performed by SOM partitions the input data space spanned by $\{\mathbf{x}\}$ into a set of regions. For each region a local model for predicting the output variable y is trained in a supervised manner. The local models can be constructed either simultaniously or after the SOM has been trained [109].

A scheme for constructing and using the local models is depicted in Fig. 4.8. The local models are formed of the data vectors falling to the Voronoi regions of the map units. If a local data set of a single map unit is considered too small, it can be augmented from the data sets of similar, close-lying prototype vectors. When using the model, a prototype vector is first chosen from the map, and then the associated local model is used for prediction.

In regression the local models are usually linear. In classification they are typically lookup-models. Using samples with known classes, the SOM is labeled

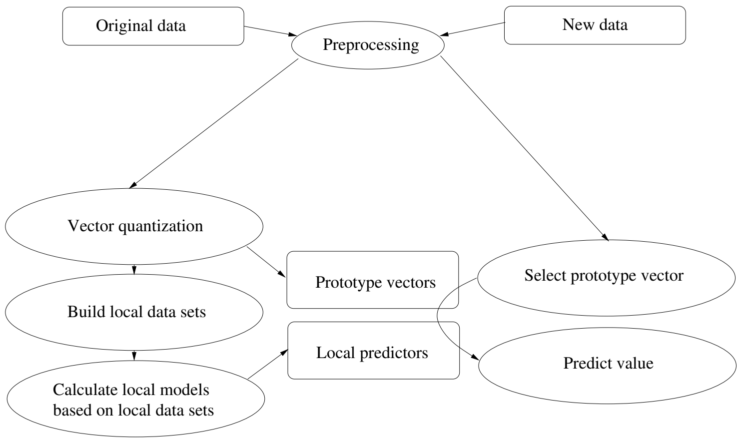


Figure 4.8: A scheme for constructing local models.

with the class identifiers, and these labels are used to classify new samples. Notice, however, that this is clearly a suboptimal solution, since the class information is in no way used in positioning the prototype vectors. There are some schemes which can be used to optimize the SOM for classification [48]. Ideally, the number of prototypes should be much larger on the cluster borders where much accuracy is needed. The Learning Vector Quantizer (LVQ) is a close relative of SOM which is designed for classification [58]. SOM can be utilized in its initalization.

Probability density estimates based on SOM are typically additive mixture models. Each map unit has an associated probability density function and an a priori probability. By adding these in a Bayesian manner, an additive mixture model is obtained [3, 36, 43]. Alternatively, the SOM can be utilized in construction of hidden Markov models [69].

Chapter 5

Conclusion

In this thesis, the use of the Self-Organizing Map (SOM) in data mining has been discussed. Its main features are:

- The SOM algorithm can be applied to numerical table-format data.
- The SOM combines properties of both vector quantization and vector projection algorithms, and thus provides
  - a low-dimensional map of the data manifold which tunes to the local data density and can be utilized efficiently in visualization.
  - a robust set of prototypes which roughly follow the probability density of the training data and can thus be used for clustering and analysis of the data.
- If the number of map units is chosen proportional to $\sqrt{N}$, the computational complexity of training scales linearly with the size of data $\mathcal{O}(Nd)$, where N is the number of data samples and d is the input dimension.

Several ways to make use of the SOM have been suggested. Basically, the SOM is a visualization, clustering and projection tool. It is found to be especially suitable for data understanding, but it can also be used for data preparation and, combined with local models, for modeling and probability density estimation. As such, it is a prime tool in preparation-survey –cycle introduced in Section 2.7.

An outline of the usage of the SOM in the preperation-survey –cycle is presented in Fig. 4.1. The main steps are data preparation, SOM training, visualization, clustering, summarization and modeling. The publications deal with visualization [110], clustering [112] and modeling [111] steps. In addition, this thesis briefly discusses the summarization step. Future work will concentrate on this step to complete the loop in Fig. 4.1, and thus make efficient use of the SOM in the preparation-survey –cycle possible.

The SOM training algorithm, as well as many of the methods mentioned in this thesis have been implemented in the SOM Toolbox [113, 114] for Matlab 5 computing environment. The SOM Toolbox can be downloaded free of charge from `http://www.cis.hut.fi/projects/somtoolbox/`. Hopefully, this work will inspire other researchers and fellow data engineers to make use of the potential of the SOM as a versatile data mining tool.

Bibliography

[1] *Proceedings of Workshop on Self-Organizing Maps (WSOM'97)*, Espoo, Finland, 1997. Helsinki University of Technology, Neural Networks Research Centre.

[2] D. Alahakoon and S. K. Halgamuge. Knowledge Discovery with Supervised and Unsupervised Self Evolving Neural Networks. In Yamakawa and Matsumoto [117], pages 907–910.

[3] Esa Alhoniemi, Johan Himberg, and Juha Vesanto. Probabilistic Measures for Responses of Self-Organizing Map Units. In Bothe et al. [10], pages 286–290.

[4] Esa Alhoniemi, Jaakko Hollmén, Olli Simula, and Juha Vesanto. Process Monitoring and Modeling Using the Self-Organizing Map. *Integrated Computer-Aided Engineering*, 6(1):3–14, 1999.

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

[6] James C. Bezdek. Some New Indexes of Cluster Validity. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 28(3):301–315, 1998.

[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 Recogition*. Oxford University Press, 1995.

[9] 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.

[10] H. Bothe, E. Oja, E. Massad, and C. Haefke, editors. *Proceeding of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA'99)*. ICSC Academic Press, 1999.

[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. [27], chapter 2: The Process of Knowledge Discovery in Databases.

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

[14] 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.

[15] Basabi Chakraborty and Yasuji Sawada. Feature selection by artificial neural network. In Yamakawa and Matsumoto [117], pages 724–727.

[16] Pete Chapman, Julian Clinton, Thomas Khabaza, Thomas Reinartz, and Rüdiger Wirth. The CRISP-DM process model. Technical report, CRISM-DM consortium, March 1999. A discussion paper available from http://www.crisp-dm.org.

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

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

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

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

[21] 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.

[22] 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.

[23] Inderjit S. Dillon, 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.

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

[25] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.

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

[27] 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.

[28] 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.

[29] Bernd Fritzke. Let it grow – self-organizing feature maps with problem dependent cell structure. In Kohonen et al. [56], pages 403–408.

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

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

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

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

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

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

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

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

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

[39] 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.

[40] 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.

[41] Johan Himberg. A SOM based cluster visualization and its application for false coloring. Accepted for publication in International Joint Conference in Neural Networks (IJCNN), 2000.

[42] Timo Honkela, Samuel Kaski, Krista Lagus, and Teuvo Kohonen. WEBSOM—self-organizing maps of document collections. [1], pages 310–315.

[43] Jukka Iivarinen. *Texture Segmentation and Shape Classification with Histogram Techniques and Self-Organizing Maps.* PhD thesis, Helsinki University of Technology, 1998. Acta Polytechnica Scandinavica: Mathematics, Computing and Management in Engineering, 95.

[44] 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.

[45] John F. Elder IV and Daryl Pregibon. Advances in knowledge discovery and data mining. In Fayyad et al. [27], chapter 4: A Statistical Perspective on Knowledge Discovery in Databases, pages 83–113.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] Samuel Kaski, Jarkko Venna, and Teuvo Kohonen. 14: Tips for Processing and Color-Coding of Self-Organizing Maps. In Deboeck and Kohonen [20], pages 195–202.

[53] 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.

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

[55] K. Kiviluoto. Comparing 2D and 3D self-organizing maps in financial data visualization. In Yamakawa and Matsumoto [117], pages 68–71.

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

[57] Teuvo Kohonen. Self-Organizing Maps: Optimization Approaches. In Kohonen et al. [56], pages 981–990.

[58] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences.* Springer, Berlin, Heidelberg, 1995.

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

[60] 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.* Accepted for publication.

[61] 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.

[62] 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.

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

[64] Pasi Koikkalainen and Erkki Oja. Self-organizing hierarchical feature maps. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'90)*, pages 279–284, 1990.

[65] Andreas König. A survey of methods for multivariate data projection, visualization and interactive analysis. In Yamakawa and Matsumoto [117], pages 55–59.

[66] Bart Kosko. *Fuzzy Engineering.* Prentice Hall International, 1997.

[67] 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.

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

[69] Mikko Kurimo. *Using Self-Organizing Map and Learning Vector Quantization for Mixture Density Hidden Markov Models.* PhD thesis, Helsinki University of Technology, 1997. Acta Polytechnica Scandinavica: Mathematics, Computing and Management in Engineering, 87.

[70] Krista Lagus, Timo Honkela, Samuel Kaski, and Teuvo Kohonen. Self-organizing maps of document collections: A new approach to interactive exploration. In Evangelios Simoudis, Jiawei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 238–243. AAAI Press, Menlo Park, California, 1996.

[71] 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.

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

[73] 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.

[74] S. P. Luttrell. A Bayesian analysis of self-organising maps. *Neural Computation*, 6(5):767–794, 1994.

[75] 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.

[76] 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.

[77] 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.

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

[79] 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.

[80] 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.

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

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

[83] Janne Nikkilä. Itseorganisoivan kartan tietoaineistosta muodostaman kuvauksen tulkintamenetelmiä. Master's thesis, Helsinki University of Technology, 1999.

[84] W. Pedrycz and H. C. Card. Linguistic interpretation of self-organizing maps. In *Proceedings of International Conference on Fuzzy Systems '92*, pages 371 – 378, 1992.

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

[86] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 143–167. Clarendon Press, Oxford, 1987.

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

[88] 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.

[89] Marina Resta. Self organizing evolutionary models in financial markets forecasting. [1], pages 187–190.

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

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

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

[93] 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.

[94] T. Samad and S. A. Harp. Feature map learning with partial training data. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'91)*, volume II, page 949, Piscataway, NJ, 1991. IEEE Service Center.

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

[96] 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.

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

[98] 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.

[99] Olli Simula, Juha Vesanto, Esa Alhoniemi, and Jaakko Hollmén. *Neuro-Fuzzy Techniques for Intelligent Information Systems (N. Kasabov and R. Kozma, eds.)*, chapter 1: Analysis and Modeling of Complex Systems Using the Self-Organizing Map. Springer, 1999.

[100] Olli Simula, Juha Vesanto, and Petri Vasara. Analysis of Industrial Systems Using the Self-Organizing Map. In L. C. Jain and R. K. Jain, editors, *Proceedings of the Second International Conference on Knowledge-Based Intelligent Electronic Systems*, pages 61–68. IEEE, 1998.

[101] 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.

[102] 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.

[103] 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.

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

[105] A. Ultsch. Self organized feature maps for monitoring and knowledge acquisition of a chemical process. In S. Gielen and B. Kappen, editors, *Proceedings of International Conference on Artificial Neural Networks (ICANN'93)*, pages 864–867, London, UK, 1993. Springer.

[106] 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.

[107] A. Varfis and C. Versino. Clustering of Socio-Economic Data with Kohonen Maps. *Neural Network World*, 2(6):813–834, 1992.

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

[109] Juha Vesanto. Using the SOM and Local Models in Time-Series Prediction. [1], pages 209–214.

[110] Juha Vesanto. SOM-Based Data Visualization Methods. *Intelligent Data Analysis*, 3(2):111–126, 1999.

[111] Juha Vesanto and Jussi Ahola. Hunting for Correlations in Data Using the Self-Organizing Map. In Bothe et al. [10], pages 279–285.

[112] Juha Vesanto and Esa Alhoniemi. Clustering of the Self-Organizing Map. *IEEE Transactions on Neural Networks*. Accepted for publication.

[113] Juha Vesanto, Esa Alhoniemi, Johan Himberg, Kimmo Kiviluoto, and Jukka Parviainen. Self-Organizing Map for Data Mining in Matlab: the SOM Toolbox. *Simulation News Europe*, (25):54, March 1999.

[114] 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.

[115] Juha Vesanto, Johan Himberg, Markus Siponen, and Olli Simula. Enhancing SOM Based Data Visualization. In Yamakawa and Matsumoto [117], pages 64–67.

[116] 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.

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

[118] 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.

[119] 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.

Errata

This section lists the changes made to the manuscript since 5th of April, 2000. These changes are purely corrections: they do not change the qualitative content of the thesis in any way.

- On April 10th, 2000, the C-code in Fig. 3.4 was changed slightly. In C-language the operator `'^'` is XOR, not power, so `di += (x[k] - m[i][k])^2;` was changed to `{ f = x[k] - m[i][k]; di += f*f; }`.
- On April 17th, 2000, in Section 5, a reference to [111] was changed to [3], and a few corrections in the Bibliography were made: the paper by Sammon [95] and the PhD thesis of Healey [38] were misplaced in the alphabetical list.