Interactive Visualization of Statistical Data using Multidimensional Scaling Techniques

Mattias Jansson & Jimmy Johansson

2003-02-26
Interactive Visualization of Statistical Data using Multidimensional Scaling Techniques

Examensarbete utfört i Medieteknik vid Linköpings Tekniska Högskola, Campus Norrköping

Mattias Jansson & Jimmy Johansson

Handledare: Mikael Jern och Robert Treloar
Examinator: Mikael Jern

Norrköping den 26 februari 2003
Interactive Visualization of Statistical Data using Multidimensional Scaling Techniques

Mattias Jansson & Jimmy Johansson

This study has been carried out in cooperation with Unilever and partly with the EC founded project, Smartdoc IST-2000-28137.

In areas of statistics and image processing, both the amount of data and the dimensions are increasing rapidly and an interactive visualization tool that lets the user perform real time analysis can save valuable time. Real time cropping and drill-down considerably facilitates the analysis process and yield more accurate decisions.

In the Smartdoc project, there has been a request for a component used for smart filtering in multidimensional data sets. As the Smartdoc project aims to develop smart, interactive components to be used on low-end systems, the implementation of the self-organizing map algorithm proposes which dimensions to visualize.

Together with Dr. Robert Treloar at Unilever, the SOM Visualizer - an application for interactive visualization and analysis of multidimensional data - has been developed. The analytical part of the application is based on Kohonen’s self-organizing map algorithm. In cooperation with the Smartdoc project, a component has been developed that is used for smart filtering in multidimensional data sets. Microsoft Visual Basic and components from the graphics library AVS OpenViz are used as development tools.

This study focuses on developing an application, which enables extensive real time interaction techniques based on Kohonen’s self-organizing map algorithm.
Interactive Visualization of Statistical Data using Multidimensional Scaling Techniques

Mattias Jansson
Jimmy Johansson

2003-02-26
ABSTRACT

This study has been carried out in cooperation with Unilever and partly with the EC founded project, Smartdoc IST-2000-28137.

In areas of statistics and image processing, both the amount of data and the dimensions are increasing rapidly and an interactive visualization tool that lets the user perform real-time analysis can save valuable time. Real-time cropping and drill-down considerably facilitate the analysis process and yield more accurate decisions.

In the Smartdoc project, there has been a request for a component used for smart filtering in multidimensional data sets. As the Smartdoc project aims to develop smart, interactive components to be used on low-end systems, the implementation of the self-organizing map algorithm proposes which dimensions to visualize.

Together with Dr. Robert Treloar at Unilever, the SOM Visualizer - an application for interactive visualization and analysis of multidimensional data - has been developed. The analytical part of the application is based on Kohonen’s self-organizing map algorithm. In cooperation with the Smartdoc project, a component has been developed that is used for smart filtering in multidimensional data sets. Microsoft Visual Basic and components from the graphics library AVS OpenViz are used as development tools.
# Table of Contents

1. **Introduction** - 1
   1.1 **Background** - 1
   1.2 **Objectives of Project** - 1
   1.3 **Method** - 1

2. **Fundamentals of Report** - 3
   2.1 **Objectives of Report** - 3
   2.2 **Audience** - 3
   2.3 **Structure of Report** - 3

3. **Information Visualization** - 4
   3.1 **Visualization Techniques** - 4
   3.2 **Interaction Techniques** - 5

4. **Graphical User Interface** - 7

5. **Programming Language and Development Tools** - 8
   5.1 **COM** - 8
   5.2 **Microsoft Visual Basic** - 8
   5.3 **AVS OpenViz** - 9
   5.4 **Choice of Programming Language and Development Tools** - 9

6. **Simplification of Multidimensional Data Sets** - 11
   6.1 **Clustering** - 11
   6.2 **Projection Methods** - 12
      6.2.1 **Linear Projection Methods** - 12
      6.2.2 **Nonlinear Projection Methods** - 13
   6.3 **Self-Organizing Map** - 15

7. **Choice of Algorithm for Implementation** - 18

8. **Implementation** - 20
   8.1 **Self-Organizing Map Algorithm** - 20
   8.2 **Graphical User Interface** - 22
      8.2.1 **Neuron Values Window** - 23
      8.2.2 **Advanced Mode** - 24
   8.3 **Visual User Interface** - 25
      8.3.1 **Drill-Down** - 25
      8.3.2 **Rectangle Manipulator** - 26
# Table of Contents

8.3.3 CROPPING .............................................. 26
8.4 VISUALIZATION COMPONENTS ......................... 27
  8.4.1 PARALLEL COORDINATE PLOTS .................. 27
  8.4.2 PATCH CHARTS ...................................... 27
  8.4.3 SURFACE MAP ...................................... 28
  8.4.4 COLOR MAPPING .................................... 29
8.5 READ COMPONENT ........................................ 30
8.6 WORKFLOW .............................................. 30
8.7 SMARTDOC SCATTER COMPONENT ....................... 32

9 INNOVATIONS .................................................. 34

10 ASSESSMENT AND FUTURE WORK ......................... 36

11 CONCLUSION ................................................ 38

12 ACKNOWLEDGEMENTS ...................................... 40

13 ABBREVIATIONS ............................................ 41

14 GLOSSARY .................................................. 42

15 REFERENCES ............................................... 43

16 APPENDIX A ................................................ 45

17 APPENDIX B ................................................ 48
LIST OF FIGURES

FIGURE 3-1 – PATCH CHART WITH TWO DIFFERENT COLORMAPS - 4 -
FIGURE 3-2 – SURFACE MAP - 5 -
FIGURE 3-3 – PARALLEL COORDINATE PLOT - 5 -
FIGURE 5-1 – SCREENSHOT FROM MS VISUAL BASIC 6.0 - 9 -
FIGURE 6-1 – DENDROGRAM - 12 -
FIGURE 8-1 – SOM VISUALIZER - 23 -
FIGURE 8-2 – NEURON VALUES WINDOW - 24 -
FIGURE 8-3 – SOM VISUALIZER ADVANCED MODE - 25 -
FIGURE 8-4 – RECTANGLE MANIPULATOR - 26 -
FIGURE 8-5 – CROPPING IN PARALLEL COORDINATE PLOT - 26 -
FIGURE 8-6 – WORKFLOW IN SOM VISUALIZER - 31 -
FIGURE 8-7 – WORKFLOW IN SMARTDOC SCATTER COMPONENT - 32 -
## LIST OF EQUATIONS

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equation 6-1</td>
<td>Stress Function</td>
<td>14</td>
</tr>
<tr>
<td>Equation 6-2</td>
<td>Sammon Mapping Stress Function</td>
<td>15</td>
</tr>
<tr>
<td>Equation 6-3</td>
<td>Gradient Descent Algorithm</td>
<td>15</td>
</tr>
<tr>
<td>Equation 8-1</td>
<td>City Block</td>
<td>20</td>
</tr>
<tr>
<td>Equation 8-2</td>
<td>Euclidean</td>
<td>20</td>
</tr>
<tr>
<td>Equation 8-3</td>
<td>8-Connectivity</td>
<td>20</td>
</tr>
<tr>
<td>Equation 8-4</td>
<td>Excitation &amp; Inhibition</td>
<td>21</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

This report describes the master thesis *Interactive visualization of statistical data using multidimensional scaling techniques*. It has been carried out in cooperation with Unilever and partly with the EC founded project, Smartdoc IST-2000-28137.

1.1 Background

As a result of the rapid development in the field of information technology, demand has increased on accurate evaluation of data. Research areas e.g. neural networking, knowledge discovery and data mining are continuously becoming more significant to fulfill those needs. When the amount of data increases, both in terms of size and dimensions, it is becoming harder to make accurate interpretations that still retain the main features of the data. Consequently, there is a need for new ways of handling multidimensional data.

1.2 Objectives of Project

The two main objectives of this project are to develop a tool for analysis of multidimensional data sets, which enables extensive real-time interaction techniques in visualizations, and to develop an algorithm based on a similarity criterion for classification of dimensions to be integrated in a Smartdoc scatter component.

Both the algorithm for classification of dimensions and the tool for analysis of multidimensional data sets (henceforth referred to as the SOM Visualizer) are to be based on a scaling algorithm selected in an initial theoretical study.

The development of the algorithm for integration in the Smartdoc scatter component should focus on automatic selection of the dimensions an expert would choose. The development of the SOM Visualizer should focus on first implementing the chosen scaling algorithm in order to develop a program that supports interactive visualization of multidimensional data sets. Consequently, the techniques for visualization and interaction are expected to be the most mature parts of the program.

1.3 Method

The project consisted of two parts, a theoretical and a practical. In the former, the field of scaling techniques was studied. The duration of the evaluation process was four weeks and ended up with the choice of algorithm for implementation. The implementation spanned over sixteen weeks during which both the stand-alone application and the classification algorithm for the Smartdoc scatter component were developed.

The gathering of information for the choice of algorithm for implementation mainly focused on the study of research papers in the field of dimensional scaling. Different approaches were taken into consideration, but requests from Unilever made the choice of algorithm depend primarily on the possibilities to create flexible implementations of the algorithms practically.
Feedback has played an important role during the development process of the project as our examiner Mikael Jern and Robert Treloar at Unilever frequently have commented the work. Also, when a beta version of the SOM Visualizer had been developed, neural networking experts at Unilever and visualization specialists at the department of science and technology at Linköping University have performed beta testing on the application.
2 FUNDAMENTALS OF REPORT

2.1 Objectives of Report
The first objective of this report is to show that by tightly coupling the self-organizing map algorithm with new combinations of visualization components and techniques for interactivity, real-time analysis of multidimensional data can be made.

The second objective is to describe how the self-organizing map algorithm can be used as a base for classification of dimensions in a multidimensional data set.

2.2 Audience
The audience for this report is primarily partners at Unilever Research, collaborators in the Smartdoc project and Linköping University.

2.3 Structure of Report
Chapter three gives an introduction to the research areas of information visualization and briefly describes the most common visualization and interaction techniques. In chapter four, the focus is set on graphical user interfaces. Readers familiar with the latter two fields can ignore these sections. Chapter five describes the technical foundations on which this project is based. Chapter six gives a thorough theoretical study about existing scaling algorithms. It provides the background needed for the choice of algorithm described in chapter seven. Chapter eight then describes the implementation phase and chapter nine contains a summary of the innovative aspects of the project. Chapter ten finally deals with the evaluation and assessment of the application along with ideas of possible future work. Explanations of topic specific terminology and abbreviations are given in the last sections of the report.
3 INFORMATION VISUALIZATION

During the last decade, a new visualization research focus has emerged, intended to visualize abstract information. Information visualization transforms abstract information into visual information enabling all the advantages of quick insight and pattern recognition that are intrinsic in visual information. Information visualization has its roots in scientific visualization, one of the first research areas in which computers were used on a large scale for visualization purposes. Consequently, there are many similarities between the two disciplines. However, one difference between scientific and information visualization is that information visualization focuses on abstract data.

Mapping this abstract data to the physical world is very difficult, or as in most cases, almost impossible. Hence, a key research area in information visualization is to discover new visualization metaphors for representation of information and to understand what analysis task they support. Another difference between scientific visualization and information visualization is their audience. Information visualization has a more diverse audience in contrast to the more specialized, technical users of scientific visualization. Many occupational groups use, or are likely to use applications based on information visualization technology in the near future. A few examples of these occupational groups could be stockbrokers, chemists, statisticians and just about every scientist who has a large amount of data.

3.1 Visualization Techniques

In the area of information visualization, different techniques are used to visualize data. If the data is two-dimensional and structured, a patch chart can be used. A patch chart consists of a grid of patches that are colored and positioned according to data values. This visualization technique gives a good understanding of the data clusters, see figure 3-1.

A surface map provides a three-dimensional display of the data. Using a colormap and assigning the third dimension to a specific color produces a very intuitive map. The advantage of a surface map is that height differences easily can be recognized, see figure 3-2.
When the data becomes multidimensional, other visualization techniques, such as the parallel coordinate plot, are needed for visualization. The primary advantage of the parallel coordinate plot is its ability to display multidimensional data in one representation, breaking the traditional bounds of two- or three-dimensional multivariate representations such as scatter plots. A scatter plot effectively reveals relationships or association between two or three variables [1]. Each observation in a data set is represented as an unbroken series of line segments, which intersect vertical axes, each scaled to a different variable [2]. The value of the variable for each observation is plotted along each axis relative to the minimum and maximum values of the variable for all observations; the points are then connected using line segments. Observations with similar data values across all variables share similar signatures. Clusters of similar observations can thus be discerned, see figure 3-3.

**3.2 Interaction Techniques**

When the data is visualized, it is desirable not only to observe, but also to manipulate with the result. Many ways of interaction with visualization exist. Two of the most common interaction techniques are drill-down [3] and cropping.
By clicking on a particular data element the user is able to drill down and get more detailed information. This technique makes it possible to get an overview and if desirable, detailed information about any data element in the visualization.

When the data set visualized is large, it is sometimes preferable to view just a selected region of it and this can be done by cropping the data. There are several different cropping techniques and which one to use depends on the visualization method used. If a parallel coordinate plot is used the user can use different types of arrows to encircle parts of the data. In a three-dimensional scatter plot, several planes can be used as a bounding box. Cropping is an efficient and easy technique for reducing rendering and calculation time when dealing with large data sets. Also, if the user is especially interested in a part of a data set, cropping can be used and the visualization view can be used to display just the cropped part. This gives a better resolution and facilitates interpretation of the data.
4 Graphical User Interface

When designing a graphical user interface (GUI) it is important to have basic knowledge in the human computer interaction (HCI) field.

The short-term memory (STM) of a human being is a limited memory that acts like a buffer and has a capacity of between five and nine items. Such items can be single objects or coherent chunks of information. The size of information that can be stored in the STM depends on how familiar the user is with the environment. The STM is very unstable and distractions like external noise or other tasks quickly disrupt its content. When information has been stored five to twenty seconds in the STM it is stored in the long-term memory (LTM) [4].

The LTM is more stable and has much more capacity, but it is slower to access than the STM. A problem with the LTM is that the retrieval phase is quite difficult and mnemonic aids are used to access the stored information. [4]

A GUI should be designed in a way that makes users work as much as possible with the STM. When only access the STM the memory load of the user is lighter and the interaction is quicker and more error-free. Control and automation is another important issue in user interface design. In most cases it is useful to provide the automation of some features, but this will take away control. It is important that the user has a sense of control of the application because people tend to become frustrated when they do not have full control of their work. A critical factor for successful GUI implementation is the balance between automation and control; to show meaningful details and hide the rest depending on what task the user is performing [5].

Below are a few of the general principles that should be kept in mind when designing a graphical user interface.

- **Know your user.** This is perhaps the single most important factor when constructing a user interface, though it is sometimes very difficult to foresee whom the end-users would be.

- **Minimizing the load on users.** By providing informative feedback, memory aids and other cognitive support, the memory and cognitive load can be reduced.

- **Preserving consistencies.** There are many consistencies to be preserved in a user interface e.g. labeling, terminology, components, and layout.

- **Following standards.** There are many standards and guidelines for interactions, abbreviations and terminology. They ensure high quality while reducing the design effort.
5 PROGRAMMING LANGUAGE AND DEVELOPMENT TOOLS

The two main tools used for development of the application are Microsoft Visual Basic 6 and AVS OpenViz 2.2.

5.1 COM
COM is an acronym for Component Object Model. It was created by Microsoft to give the ability to put code written in any programming language into a package called a component to be used by one or many clients, regardless of the programming language used on the client. COM uses the fact that the clients do not need to know how the components they are using achieve their results, but only how to interpret their output [6]. COM accesses the functionality via so-called interfaces. COM technology is used in different situations such as regular applications and applications for the web. The only thing they have in common is that they all take advantage of the flexibility, extensibility and reusability COM offers. Examples of COM based technologies are ASP, in which all intrinsic objects are COM objects and Microsoft’s Internet Information Server (IIS). Almost every single Microsoft product is based on COM, and so are many third-party products. In this project, the application built was created using the COM version of AVS OpenViz.

5.2 Microsoft Visual Basic
In Microsoft Visual Basic the "Visual" part refers to the method used to create the GUI. Instead of writing numerous lines of code to describe the appearance and position of interface elements, the prebuilt objects are used to put elements on the screen. Visual Basic has evolved from the original BASIC language and now contains numerous statements, functions, and keywords, many of which relate directly to the Windows GUI. The Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system, Applications Edition included in Microsoft Excel, Microsoft Access, and many other Windows applications use the same language. The Visual Basic Scripting Edition (VBScript) is a widely used scripting language that is a subset of the Visual Basic language [7]. For a screenshot from the version of Visual Basic used, see figure 5-1.
5.3 AVS OpenViz

OpenViz is developed by Advanced Visual Systems (AVS) and includes a powerful graphics display system including data selection and interaction tools. OpenViz has identical architecture in both COM and JavaBean standards. OpenViz Java version runs on any machine with a Java Virtual Machine and the COM version runs on all Windows platforms and is compatible with Apache and Microsoft IIS web servers. OpenViz enables easy implementation of two and three dimensional visualization applications and it can be used with any development environment that supports ActiveX/COM, including Visual Basic, Visual C++ and Delphi.

5.4 Choice of Programming Language and Development Tools

The task-description of the project included the usage of AVS OpenViz and the possibility of using existing visualization components. Thus, it was clear that it would be preferable to choose a language that could easily integrate these components.

Visual Basic (VB) hides the majority of the COM runtime, which lowers the threshold of learning COM [6]. Thus, VB is the language that offers the fastest way to get started writing COM-based applications. Though not the fastest programming language available VB is estimated to be fast enough for the application. VB is also known as a fast language in terms of development time. These factors combined made VB the natural choice.
The most well known development tool for VB is Microsoft Visual Basic, one of the programs in Microsoft Visual Studio suite of tools for developing solutions. The program was available at the university. Therefore, it became the development tool of choice. The version of OpenViz available was v.2.2 which is based on the version of VB available in Visual Studio 6.0. Consequently, that version was used.
6 SIMPLIFICATION OF MULTIDIMENSIONAL DATA SETS

The more dimensions a data set contains, the harder it is to extract information such as relations between different dimensions. It is also harder to visualize the data. For visualization of more than three dimensions of data, colors, shapes and / or size etc. have to be used, or information about some dimensions has to be discarded. Sometimes it is impossible to represent all dimensions in the data set without discarding some information.

In such a situation, knowledge about the information sought is critical. If the relationship between certain dimensions is important, the values from those dimensions can be plotted. Sometimes several plots are used. In this case, all information within the plotted dimensions is kept. This is often necessary in empirical sciences. In experimental sciences such as psychology, genetics and linguistics, however, the relationship between the data elements is the important thing. The data dealt with in this project belongs to the latter category.

To be able to visualize a huge data set in a high-dimensional space effectively in a low-dimensional space with requirements regarding speed and preservation of important relations, two things need to be done. First, the number of data elements has to be reduced. Here, clustering techniques come in handy. Also, the number of dimensions has to be reduced to fit the output format. For this, several projection methods are available.

6.1 Clustering

The goal of clustering is to reduce the amount of data by categorizing or grouping similar data items together. The two main techniques of clustering are hierarchical and partitional clustering.

Hierarchical clustering can be done in two different ways. The first method is to merge smaller clusters into bigger ones according to some criteria. The other method does the opposite; it splits larger clusters into smaller ones. The result of the clustering algorithm is a tree of clusters called a dendrogram. The dendrogram is a representation of how the clusters are related. By cutting the dendrogram at a desired level, a clustering of the data items into disjoint groups is obtained. In the example dendrogram shown in figure 6-1 the threshold is set to two.
6 Simplification of Multidimensional Data Sets

Partitional clustering attempts to decompose the data into a set of disjoint clusters. The clustering algorithm tries to minimize a criterion function. This criterion usually involves minimization of some measure of similarity in the samples within each cluster, while maximizing the dissimilarity of different clusters. Several methods for partitional clustering exist and one of the most commonly used is the K-means clustering algorithm. The criterion function for the K-means clustering calculates the average squared distance of the data items from their nearest cluster centroids.

6.2 Projection methods

While clustering reduces the number of data items, it preserves the number of dimensions. As a result, it is impossible to visualize a clustered high-dimensional data set without reducing the dimensionality. During the years, several methods have been developed to preserve data structures, each optimized to preserve different aspects of those structures.

The projection methods can be divided into two different groups, linear and nonlinear projection methods. Principal Component Analysis (PCA) and Projection Pursuit belong to the former group while Multidimensional Scaling (MDS) and Principal Curves belong to the latter.

6.2.1 Linear Projection methods

The most well known linear projection method is PCA, which uses eigenanalysis [8] to find the linear projection that best preserves the variance of the data. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The second principal component corresponds to the second eigenvector etc. The number of components chosen corresponds to the number of dimensions in the projection.

PCA is a standard method in data analysis and effective algorithms already exist. Though really effective, its linearity is a restriction to the choice of projection for multivariate proximity data, since the relative position of the data items is the
important thing while the absolute position is not. Therefore, linear projection methods are less useful for the problems faced in the particular case of this project and consequently linear approaches will not be further discussed. The reason the methods of PCA are described and those of Projection Pursuit and other linear projection methods are not, is that there exists a nonlinear projection method based on PCA, namely Principal Curves.

6.2.2 Nonlinear Projection methods
Nonlinear projections methods are able to take nonlinear structures into account when searching for the optimal projection from the n-dimensional data set to the m-dimensional properties of the output device. Consequently, using a method from this group will give a good result if the proximity relationships are crucial to preserve. The most well-known methods are presented below.

6.2.2.1 Principal Curves
Principal curves are generalizations of principal components extracted using PCA in the sense that a linear principal curve is a principal component [9]. Principal curves are smooth curves that are defined by the property that each point of the curve is the average of all data points that project to it. [10] Being a generalization of PCA, Principal Curves are more computationally intensive than its predecessor. Mulier and Cherkassy stated that discretized principal curves are almost equivalent to Self-Organizing Maps (SOMs) [10], an algorithm presented further down in this document.

6.2.2.2 Multidimensional Scaling
The purpose of MDS is to provide a visual representation of similarities or distances among a set of objects. MDS plots, for example, some properties of a car that are perceived to be similar close to each other on a map (picture) and those properties that are perceived to be different from each other are plotted far away from each other. As the amount of data that is to be processed often is immense, this technique helps the user to get a good overview of the data.

With MDS, it is customary to talk about different kinds of relationships. A positive relationship between the input similarities and distances among points on a map means that the smaller the distance is between points on the map the more similar they are to each other, and vice versa. A negative relationship means the opposite: the smaller the input similarities between items, the farther apart on the map they would be.

The input data to MDS is a square, symmetric matrix which contains relationships among a set of items. A matrix can be either a similarity or a dissimilarity matrix. It is a similarity matrix if larger numbers indicate more similarity between items and it is dissimilarity matrix if the opposite applies. However, this can in some cases be inaccurate, because many input matrices are neither similarities nor dissimilarities.

A typical example of an input matrix is the aggregate proximity matrix derived from a pile sorting task. Each cell of such a matrix holds the number of respondents who placed two different items into the same pile. It is assumed that the number of respondents placing two items into the same pile is an indication of how similar they are each other. If an MDS map would be implemented with such an input matrix, the
result would be that items which were often sorted into the same pile would lie close to each other, and vice versa [11].

Another typical example of an input matrix is a matrix of correlations among a set of variables. If these data were treated as similarities, the MDS algorithm would place variables with high correlation close to each other and the variables with negative correlation far apart. The input can also be a matrix that consists of the numbers of business transactions between different corporations. Running this type of data with an MDS application might reveal clusters of corporations that trade more often with each other then others do. If the numbers of business transaction are treated as similarities, pairs of companies with many business transactions would lie closer to each other then pairs of companies with less business transactions.

One goal with MDS is to find a visual interpretation of a complex set of relationships. The visualization should give the user an overview and be easy to understand. Since ordinary maps printed on paper are two-dimensional objects, the goal of MDS would also be to find an optimal configuration of points in two-dimensional space. In reality, however, the best possible configuration of points in two-dimensional space may be a very poor and highly distorted representation of the data used [11]. When this happens there are only two alternatives: either abandon MDS as a method of representation, or increase the number of dimensions.

Once the dimensions are increased there are new difficulties to take into consideration. Even three dimensions are difficult to display on an ordinary piece of paper and are also significantly more difficult to comprehend. Going up to four or more dimensions makes MDS useless as a visualization method. Another problem with the increase of dimensions is that it is necessary to estimate an increasing number of parameters to obtain a decreasing improvement in stress [11], which is discussed below. The result is a model of the data that is nearly as complex as the data itself.

The degree of correspondence between the distances among points on the MDS map and the input matrix is inversely measured by a stress function. The general form of the stress function is:

\[
\sqrt{\frac{\sum \sum (f(x_{ij}) - d_{ij})^2}{\text{scale}}}
\]

(Equation 6-1)

\(d_{ij}\) is the Euclidean distance, across all dimensions, between points i and j on the map. \(F(x_{ij})\) is a function of the input data, and scale is a constant that is used to keep the stress value between 0 and 1. When the MDS map perfectly reproduces the input data, \(f(x_{ij}) - d_{ij} = 0\), i.e. the stress is zero [11].

Mathematically, non-zero stress can only occur if an insufficient number of dimensions is used. This implies that for any given data set, it may be impossible to perfectly represent the input data with two or three dimensions. On the other hand, any type of data set can be perfectly represented using \(n-1\) dimensions [11], where \(n\)
is the number of items scaled. This does not mean that an MDS map need to have zero stress in order to be useful, because a small amount of distortion is acceptable.

### 6.2.2.3 Sammon Mapping

Often, the Euclidian distances in the mapped data set may deviate from the original as long as the rank order is preserved better than in metric scaling. In such cases, Sammon mapping is a good choice. Just as the methods discussed above, Sammon mapping uses optimization of a cost function.

\[
E_i = \sum_{k \neq l} \frac{[d(k,l) - d'(k,l)]^2}{d(k,l)}
\]  

(Equation 6-2)

The division with the original space emphasizes the preservation of small distances. This is a result of the normalization process (each squared distance error is divided by the original distance) that makes Sammon mapping different from other MDS algorithms. Given the error function above, an optimal projection can be computed using a gradient descend algorithm.

\[
x_{ij}^{t+1} = x_{ij}^t + \alpha \frac{\partial E(t)}{\partial x_{ij}^t}
\]  

(Equation 6-3)

### 6.2.2.4 Other methods

Because nonlinear MDS methods are slow for large data sets, some simplified methods have been developed. Generally these methods are less time-consuming and used under the right circumstances they generate approximately the same result as algorithms taking all aspects into account. Usage of a simplified algorithm requires good knowledge of the data and about the questions asked about it, or the underperformance in accuracy may be substantial.

One simplification is to restrict the attention to a subset of the distance matrix. This is what the triangulation method does. It maps the points sequentially onto the plane, and the distance from the new item to the two nearest items already mapped is preserved [10]. Another method is to map the points in such an order that all of the nearest-neighbor distances will be preserved. It is also possible to reduce computation time by reducing the dimensionality of the data set. Associative neural networks have been used for this purpose but with limited success.

### 6.3 Self-Organizing Map

Kohonen’s Self-Organizing Map (SOM) has its origin in neural networking and was created as a mathematical model of the mapping between the nerve sensory and the cerebral cortex [12]. SOM is primarily used as a dimension reduction tool and as an abstraction process to represent data points with fewer representatives. The mapping is topology preserving, i.e. it forms a locally correct projection while its distance mapping globally is a mere consequence of the local projection. In this aspect, it differs from nonmetric MDS, which tries to preserve the rank order of the distances equally.
It has shown that SOM is an optimal vector quantizer in minimizing the mean-square error between the original and the mapped distances. The characteristics of SOM are useful in a wide variety of areas and it has become the most popular artificial neural network algorithm. SOM is used as a standard analytical tool in for example statistics, signal processing, control theory, financial analysis, experimental physics, chemistry and medicine.

The SOM algorithm uses competitive learning to let its neurons adapt to the input so that they eventually specialize into sensing different input stimulus [13]. The learning process is unsupervised, i.e. it does not depend on training in order to become useful. The winning neuron is “rewarded” with becoming more like the sample vector. If order among the neurons is desired, not only the winning neuron, but also its neighbors should learn from the sample vector. The neighbors do not adjust to the sample vector to the same extent though. How much a weight-vector, i.e. the vector consisting of values in all dimensions in the original space, of the neuron is adjusted depends on its closeness to the winning neuron.

When running a SOM algorithm, the output depends on a wide range of factors. First, the number of neurons has to be decided. The more neurons, the less information is lost to the cost of increased computation time. Other properties that have to be determined are metrics, the update rate of the neighbor neurons and how the learning rate varies over time.

Below, the basic principles of the SOM algorithm are described [14].

- Decide (using e.g. PCA) which plane to use for the neurons
- Set (e.g. randomize) initial x- and y-values of the neurons (scaling to two dimensions)

FOR (all vectors)
    FOR (all neurons)
        Compare the input vector with the current neuron (using some metrics)
        Save the result in a distance matrix
        Also, check if the current neuron is the best candidate so far
    END
    Now, we have a winning neuron.
    FOR (all neurons)
        Calculate neighbor distance
        IF (distance < threshold)
            Update neuron position based on distance to winner and learning rate
        END
    END
END

Though the SOM algorithm can be run on any data set, the way it works makes it more or less suitable for different categories of data.
The algorithm faces problems when the input data is based on representations, such as 1 represents the purchase department, 2 represents the sales department and 3 the delivery department. The problem is that department 1 and 3 are as similar as department 1 and 2. The SOM algorithm assumes similar numbers mean similar input data, and consequently, it is probable that similar numbers are grouped together.

When it comes to discrete data, the SOM algorithm does a good job though the results can be a bit confusing to interpret. For example, if there is a neuron having a weight vector of 12.8, it should be interpreted as the average value of the input data items probably are around 12.8 while the median probably is 13 or another integer nearby.

Clearly, the performance of the SOM algorithm is best when continuous data is used.
7 Choice of Algorithm for Implementation

The choice of scaling and clustering methods to be used when dealing with multidimensional data depends on a number of factors. The most important ones include what kind of information that is sought, the size of the data set, the need for fast or even real-time processing, the number of dimensions to be visualized, the estimated time for development of the algorithm, the ability to deal with the current data (e.g. spread or clustered data), the ability to preserve the topology, the inter-cluster relationships etc.

When it comes to scaling, there are three candidate groups: the linear projection methods e.g. PCA, the non-linear MDS methods such as Sammon mapping and the neural network SOM that also clusters the data.

PCA is not only one of the first scaling methods but also an optimal variance preserving linear projection method. Thus, it has become a well-known and often implemented algorithm. However, it is usually not adaptive and requires the entire data set. The biggest drawback is that it is unable to capture nonlinear relationships defined by higher than second order statistics [12]. The ability to preserve all relationships, linear or not, is crucial for the data. Consequently, PCA cannot be considered as a candidate algorithm. Neither are the nonlinear extensions of PCA, e.g. principal curves and principal surfaces possible candidates as they all lack a valid algorithm.

The nonlinear MDS methods are more flexible than PCA. Also, they are all good at preserving inter-cluster relationships. Unfortunately, they all suffer from vulnerability to local minima during the minimization of the stress function and are often computationally intensive. Also, they all lack an explicit projection function. Thus, if new data is added, the algorithm has to be recalculated for the entire data set.

Despite its computational complexity, Sammon mapping is the most interesting non-linear MDS algorithm because it converges faster and preserves the topology better than the other algorithms that are based on a gradient method. The advantages of preserving both intra- and inter-cluster relationships make Sammon mapping a good candidate for the dimensional scaling part of the program.

SOM is one of the fastest algorithms for dimensional scaling and offers superior topological accuracy. Thus, SOM is one of the most used algorithms for dimensional scaling. On top of that, SOM allows the addition of data without reruns of the algorithm and performs clustering at the time complexity $O(n^2)$. Its biggest drawback is that SOM does not directly show inter-neuron distances and consequently SOM created maps are less reliable than e.g. Sammon mapping when comparing long distance relationships.

Research has been done on these shortcomings, and several methods have been developed to improve the graphical output of SOM. Some of the solutions focus on the visualization of the inter-neuron relationships, e.g. adding colored borders between the neurons giving closely related neighboring neurons brighter borders than
less related ones [15]. Another solution is to modify the way the neighborhood is updated. The latter is applied in ViSOM [12], one of the more successful modifications of SOM.

SOM was chosen due to the speed, the topological accuracy and the built-in clustering along with the superior amount of resources available on the topic. The time-limitation of the project and the scalability SOM offers in extensions like ViSOM and the extended Kohonen maps presented by Peter Kleirweg, [15] are other reasons why a decision was easy to make. The choice of SOM as scaling algorithm also settles the choice of clustering method, since SOM automatically clusters the data. The reason why clustering with SOM is never compared with other clustering methods is that this project focuses on scaling.
8 IMPLEMENTATION

This chapter describes the implementation phase of the project. It describes the implementation of the chosen algorithm, the SOM algorithm, the visualization components, the user interface and the Smartdoc scatter component.

In this study, the SOM algorithm has been used in two different applications with different purposes. In the SOM Visualizer, which is a standalone application, the algorithm is used together with different visualization components in order to find correlations in multidimensional data sets. Since the focus when developing the application has been to implement the SOM algorithm and visualization components that enable the best interactivity, existing theory of user interfaces has not been thoroughly studied, although the general principles of HCI have been followed.

In the Smartdoc project, the algorithm is used as a smart filter for choosing which three dimensions that best represents the multidimensional data set. These three dimensions are then visualized in a three-dimensional scatter plot.

8.1 Self-Organizing Map algorithm

Because the SOM algorithm has already been implemented in various programs, the main task was to develop visualization and interaction techniques. Code written by Kristof Van Laerhoven [16] was therefore used and modified.

The SOM Visualizer is able to handle several ways to calculate metric differences, and gives the user the possibility to choose among several different ways to calculate the learning of the neighboring neurons. For most settings, the best results have been obtained when the default options were used. If such is not the case, it is stated accordingly. Below, the SOM settings are listed:

- **Metric Function**
  This setting decides the metric to use when the winner is determined. The options are City-block, Euclidean (default) and 8-connectivity.

  \[
  \text{Cityblock} = |X - Y| + |Y - X| \quad (\text{Equation 8-1})
  \]

  \[
  \text{Euclidean} = \sqrt{X^2 + Y^2} \quad (\text{Equation 8-2})
  \]

  \[
  \text{8-connectivity} = \max(\Delta X + \Delta Y) \quad (\text{Equation 8-3})
  \]

- **Neighborhood**
  This setting decides how the neighborhood distance and the neighborhood radius are combined to alter the way the distance from the winning neuron affects the learning. The options are Triangular (default), Gaussian, and Excitation & Inhibition. A triangular neighborhood simply lets the learning
rate within the allowed neighborhood radius decrease linearly with the distance to the winning neuron. A Gaussian neighborhood has the form of a Gaussian curve as the learning decreases exponentially. A neighborhood based on Excitation & Inhibition is calculated as follows:

\[
1 - \left( \frac{N_d}{N_r} \right)^2 \times \exp \left( - \frac{N_d}{N_r} \right)^2
\]

Equation 8-4)

\[
N_d = \text{Neighborhood Distance}
\]
\[
N_r = \text{Neighborhood Radius}
\]

- **Neighbor Metric**
  Neighborhood Metric decides the metric to use when the neighborhood is calculated. It uses the same options as Metric Function.

- **Learning Rate**
  Each time an input vector has been associated to a neuron, the neighboring neurons are updated according to their distance to the winning neuron based on the neighborhood setting and the neighborhood radius. However, the number of times the neurons have won also affect the size of their learning factors. That is exactly what the learning rate does. The options are: Constant, Linear Decay (default), Square Root Decay and Exponential Decay.

- **Initial Learning Rate**
  The initial learning rate is the base on which the learning rate calculation is applied. A value around the default value (0.55) seems to minimize the error measurement calculations included in the program for most data sets and settings.

- **Neurons per Dimension**
  The size of the neuron grid is one of the options that have the biggest impact on calculation time. The fact that it also has a major impact on the result makes it the most obvious factor when it comes to the trade-off between speed and accuracy. For data sets containing less than 1000 items, neuron grids with 20 or less neurons per dimension have near real-time calculation times.

- **Neighborhood Radius**
  The neighborhood radius regulates the number of neighbors that get their positions updated because of the assigning of an input vector.

  Neighborhood Metric decides the metric to use when the neighborhood is calculated. It uses the same options as the metric function.

- **Pre-process Neuron Values**
  Normally, the algorithm assigns the initial values of the neuron grid randomly. When pre-processing of the neuron values is checked, the
algorithm makes qualified guesses about what is a good neuron grid. The original algorithm does the following:

- Assigns the neuron grid randomly
- Runs SOM and adjusts the neuron grid to the input vectors simultaneously

When Pre-process Neuron Values is selected, the procedure is:

- Assign the neuron grid
- Simulate a SOM run, but use the winners only to adjust the neuron grid
- Run SOM and adjust the neuron grid further

The pre-processing improves the accuracy with up to 40%. Despite that, pre-processing is turned off by default to decrease computation time.

### 8.2 Graphical User interface

Being a program written in Visual Basic, the GUI is taken care of at a high level, e.g. dragging and dropping add buttons and components. The biggest advantage with this system is that little time was needed to design the graphical user interface resulting in more time to spend on the functionality of the program itself. The other main advantage is that the program follows the Windows standard, a well-known interface that most people know of.

The GUI consists of a set of controls, for example, checkboxes, buttons and combo boxes, which the user can use to set different parameters. The different controls are grouped logically, i.e. all controls related to the SOM algorithm can be found under SOM settings. This way of grouping similar controls simplifies access by making it more intuitive, see figure 8-1.
8.2.1 Neuron Values Window

The SOM Visualizer has an optional window that is hidden by default. This is done primarily because there is not enough space on the computer screen to view it at all time. The window shows the relationship between the weight vectors calculated by the SOM algorithm, see figure 8-2. While the main window shows information of how the original data are related to the different clusters, the visualization in the neuron values window shows relationship between the different dimensions, they can be viewed separately without the loss of important information.
8.2.2 Advanced Mode

The SOM Visualizer starts in basic mode, which shows all necessary features for basic manipulation and interaction. Experienced users can use the advanced mode and this allows for setting numerous of algorithm parameters. Advanced mode also has the extension of different visualization types, as well as doing some pre-processing, see figure 8-3.
8.3 Visual User Interface

A visual user interface (VUI) complements the traditional GUI and gives the user the ability to manipulate the graphical objects directly in the visualization. The VUI gives the user a more active role in the visualization and enhances the performance of the components.

8.3.1 Drill-Down

A very important feature of the SOM Visualizer is the ability to drill down on the SOM. In the patch chart component, this is done by simply clicking on the different sub areas. The result of the selection, i.e. the input vectors corresponding to the selected cluster, is then shown in a parallel coordinate plot. Using the mouse pointer to move around in the patch chart is a very intuitive way and gives the user full control. The user can choose either to view just the result of the current cluster or to view the current cluster together with its most similar clusters. The number of most similar clusters can easily be adjusted to range from only one to the entire grid. By viewing, the current cluster together with some of its most similar clusters give a better overview of how the original data set is ordered. It also gives an understanding of how the SOM algorithm works, because it is no guarantee that the most similar clusters are the closest ones; although it is more likely.
8.3.2 Rectangle Manipulator

The rectangle manipulator is another important feature of the SOM Visualizer. With the rectangle manipulator, the user is able to select large or small groups of clusters in the surface map, see figure 8-4. This is an important feature because it enables the possibility to not only look at each cluster itself, but also a whole area of clusters. Since clusters close to each other often are more similar than clusters far apart, it can be useful to look at them as a group. The result of the selected clusters is then visualized in a parallel coordinate plot. The rectangle manipulator size is easily changed by grabbing one of its corners with the mouse pointer and then moving it to a desirable size.

![Figure 8-4 - Rectangle manipulator](image)

8.3.3 Cropping

When dealing with large data sets and only a part of the input data is of interests, it is sometimes preferable to run the SOM algorithm on a selection of the input data. The SOM Visualizer has this ability and the cropping is done by moving two arrows around in the parallel coordinate plot, see figure 8-5. This function has been developed by Staffan Palmberg at the department of science and technology, Linköping University.

![Figure 8-5 - Cropping in parallel coordinate plot](image)

In the case of a large data set, cropping can significantly reduce the calculation time needed by the algorithm. In the case when only a part of the input data is of interest, this feature eliminates the unwanted areas and the whole visualization area can be used for the selected parts.
8.4 Visualization Components
The SOM Visualizer uses three main visualization types, parallel coordinate plot, patch chart and surface map. The parallel coordinate plot is used for displaying the original data set while the patch charts and the surface map are used for the clustered and scaled data.

8.4.1 Parallel Coordinate Plots
In the SOM Visualizer, the parallel coordinate plot is used to display the multidimensional data, i.e. the input vectors (data items).

The first parallel coordinate plot is used to visualize the distribution of input vectors, the labels of the dimensions and their minimum and maximum values. It is also used to set requirements for being included in the data set that is sent to the SOM algorithm. The requirements are restrictions in the values of different dimensions. It works as described below:

\[ \text{If } (a(d) < i(d) < b(d)) \text{, for all dimensions, then i is included in the data set. a is the minimum value, i is the data item, d is the dimension and b is the maximum value.} \]

The second parallel coordinate plot is activated when the user selects neurons in the surface map or in the patch chart. It displays the data items that have been assigned to the region selected.

8.4.2 Patch Charts
The left patch chart in the main window shows the percentage of the data items that have been assigned to each neuron. This visualization effectively reveals the clusters of neurons. This is an important aspect; while the fact that neighboring neurons learn similar things makes the probability that they contain similar data items high. This visualization allows the user to select neurons to visualize in one of the parallel coordinate charts.

The second patch chart in the main window is used to visualize inter- and intra-neuron relationships, depending on which alternative that is selected. By default the intra-neuron standard deviation is displayed, since its values are easier to interpret than the nearest neighbor differences that display the inter-neuron relationships.

The intra-neuron standard deviation is as the name hints, a measurement of how similar the data items associated to each neuron are. However, the similarity is not based on the similarity to each other, but to the weight vector of the neuron. The possibly obvious reason is that the weight vector is what represents the input vectors in the output space. The value of each neuron in this patch chart is calculated as follows:

The sum of the squared differences between each input vector and the associated weight vector (the weight vector is the “value” of the neuron) for all dimensions is calculated. The square root of each sum is then saved. These two first steps are the actual standard deviation part of the calculation.
Then, the number of input data items that are associated with each neuron along with the sum of the square roots associated to each neuron are saved. The sum is then divided by the number of input data items associated to the current neuron. Thus, the average standard deviation for each neuron is calculated and the data can be visualized.

When it comes to interpreting the visualization, the following has to be taken into account: in the visualization, the legend is based not on absolute, but relative values. Normally, the average intra-neuron standard deviation is distributed such as many neurons have low relative values. The reason of this is that neurons in areas where few input data items have been assigned have had few possibilities to learn and consequently are relatively different to the input vectors that have been assigned to them. Data items that belong to dense clusters on the other hand, are spread out in areas containing many neurons due to the fact that neighbors learn in the SOM algorithm. These neurons have small errors because only input data items being very similar will be associated to a neuron. If they are not extremely close, neighboring neurons would probably be closer and would win the competition of that input data item.

The neighbor difference patch chart, displays the similarity between a neuron and its neighbors. If a neuron has a dark blue color, neurons neighboring it are likely to share most if its properties. Neurons colored red on the neighbor difference chart on the other hand are relatively different to its neighbors. Because of the way SOM works, dense clusters in the original data are assigned to proportionally many neurons as the neighboring neurons learn each time a data item is assigned to a neuron. Sparsely populated areas of the input range on the other hand, are represented by few neurons. Consequently, data items within a cluster generally receive better approximations than outliers as the former are more likely to be associated to a neuron very close to themselves. As a result of these characteristics, areas colored dark blue on the neighbor difference patch chart are very likely to be a part of a dense cluster.

The dimensional patch charts in the optional graphics window give in some cases a better overview of the properties of the data. The dimensional patch charts show the values of the weight vectors for each dimension (up to a maximum of nine). By comparing the shapes of the colors in these charts, it is easy to see in which neurons it is likely to find data items with interesting characteristics. It is also possible to see if any dimensions share similar attributes.

For example, if the values for one dimension are low in certain neurons and the same neurons most often have low values in another dimension, then the two dimensions are similar. If the data is to be visualized in a low dimensional output space in which only some of the dimensions can be displayed and if one of the two similar dimensions is visualized, the other one could be regarded as having approximately the same values.

**8.4.3 Surface Map**

Being able to show a three-dimensional landscape-like area, the surface map is a useful tool when it comes to quickly giving the user an overview of the properties of
the data. In the application, the surface map is used as a complement to the patch chart for showing the distribution of the input data items in the neuron grid, the neighbor differences or the intra-neuron standard deviation. The data itself will not be further discussed in this section, since the visualization is based on the same data as the patch chart respectively.

The main reasons for implementation are the swiftness in giving the user an overview of the data and the fact that making a rectangular selection of several neurons is more easily done in this type of visualization. The former is made easier by the color mapping based on the input data item frequencies of the neurons.

The surface map consists of the two-dimensional neuron grid and a third dimension, on which the “terrain” is based. By dragging the mouse it is possible to view the map from any direction in the three dimensional space.

8.4.4 Color Mapping

Color mapping is used in all charts and plots in the program. However, several different mappings are used, sometimes even in visualizations of the same type.

In the neuron percentage patch chart, the color mapping is based on two things, value and interaction. In this case, value means the relative percentage of input data items that have been associated with the neuron. Note that only the color is relative, the values in the legend are based on absolute values. The value determines the brightness of the neuron square. The hue and saturation, however, are decided based on the interaction status of the neuron; it is grey if the square has been selected and if the square belongs to the set of most similar clusters (neurons) it is red. Neurons belonging neither to the selections nor the neighborhood are blue. There are nine brightness levels for each interaction type in addition to black that is used for non-selected neurons that have no input data items associated. To simplify interpretation for the user, only neurons not selected have been included in the legend. After all, the selection and the neighborhood are intuitive enough without explanation.

The intra- and inter-neuron measurement patch chart is based on a different colormap where low values are represented by dark blue colors. The higher the value, the more green is added until it reaches the maximum value when blue starts to decrease. When the values are clear green, red is then added and then green decreases until only red is left. This applies for the highest values. This type of color map is relatively common as it is based on the frequencies of the colors. It is intuitive and natural and is ordered in the same way as, for example, the colors of a rainbow are ordered. Also to be mentioned, in the patch chart in the main window, a neuron becomes white when selected.

When it comes to the color mappings of the parallel coordinate plots, little is to be said. There is no logical way to colorize the values of the data items in such a plot. When working with relatively small datasets, it is useful to let the different vectors have different colors just to make it possible for the eye to separate them from each other. In the case of the plot dealing with the original data, it has not been taken into account because a component was used for that purpose, and in the case of the plot showing the selected data items, as a predefined color map has been used. In the
latter case, the color map was chosen because it gives the impression of contrast between the data items shown.

8.5 Read Component
The SOM Visualizer supports the MS Excel file format (.xls). This is because the end user almost exclusively works with this file format. The only limitation is that the MS Excel file format only supports files smaller than approximately 65000 rows. However, this is not a problem as the data sets used in the SOM Visualizer are considerably smaller.

The code used for reading from the MS excel file format was written by Staffan Palmberg at the department of science and technology, Linköping University.

8.6 Workflow
A simplified workflow of the SOM Visualizer is shown in figure 8-6. The illustration is divided between standard interaction and visualization and visualization interaction techniques. The standard interaction regards the SOM algorithm interaction, while the visualization and visualization interaction techniques describes the visualizations and the interactions used.

The workflow begins with reading of an MS Excel file. The SOM Visualizer then filters the metadata from the data containing the multidimensional data and stores it in an n-dimensional array. The original multidimensional data is simultaneously visualized in a parallel coordinate plot. The user can choose to select which input vectors that should be included and then run the algorithm with different parameters. The algorithm calculates a two-dimensional representation of the original n-dimensional data and visualizes it in different two-dimensional patch charts. The user can then select an area in the patch chart and view which parts of the original data that have been clustered together. This data is visualized in another parallel coordinate plot.
Figure 8-6 - Workflow in the SOM Visualizer
8.7 Smartdoc Scatter Component

In the Smartdoc project, the SOM algorithm along with an algorithm developed by the authors, are used as a smart filter for classification of dimensions. The data sets used in this application are multidimensional and the filter is used for finding which three dimensions that make a good representation of an n-dimensional data set. These three dimensions are then visualized in a three-dimensional scatter plot, see figure 8-7.

The algorithm developed is used for finding the three dimensions that best represent the original data set and is used together with the SOM algorithm. The n-dimensional original data set is first run through the SOM algorithm and is clustered to an n-dimensional m x m neuron grid.

The algorithm is based on the assumption that if \( n_1 \approx n_2 \) then \( n_2 \) is redundant. Therefore, \( n_1 \) is chosen to represent \( n_2 \). The same rule applies if \( n_3 \approx n_4 \approx n_5 \) and consequently \( n_3 \) is set to represents the whole group.

The algorithm starts by running the SOM algorithm on the multidimensional data set. This returns a two-dimensional grid of neurons. To make all dimensions of equal importance, the weight vectors of the neurons are normalized. When the normalization is finished, the similarities between the dimensions are found by the calculation of the differences between every dimension in the n-dimensional matrix, i.e. for each weight vector the values in different dimensions are compared and the sum of the differences is calculated for each possible combination of dimensions.

![Figure 8-7 - Workflow in Smartdoc scatter component](image)
Then, the results are sorted in a descending order, i.e. the most similar pair of dimensions at the top.

Now, the algorithm starts to construct groups based on the sorted similarity matrix. The most similar groups form a new group. This is done until only three groups remain. When two groups are combined and at least one group consists of at least two dimensions, a leader is selected. A leader is a dimension that has been selected to represent the other dimensions in the group. The dimension that is most similar to the other dimensions in the group becomes the leader. Consequently, the leader of the biggest group becomes the leader of the new group.

This is done until only three dimensions remain. The leader of each of these groups will be the dimensions represented in the visualization component. The output of the algorithm can easily be generalized to the finding of any number of dimensions. The only thing that needs to be done is to stop the grouping when the desired number of groups remains. In the case when the visualization is more easily interpreted for some dimensions than others, the best ways of visualization (normally the spatial techniques) are associated with the largest groups. For a complete description of the algorithm, see appendix A.
9 INNOVATIONS

This chapter describes the innovate aspects of the SOM Visualizer and the smart filtering algorithm for the Smartdoc scatter component.

While most current SOM applications are static, the SOM Visualizer enables real-time interaction. The SOM Visualizer enables several different ways for the user to perform real-time interaction.

The SOM Visualizer makes it possible not only to run the SOM algorithm on any data set available as an Excel spreadsheet, but also on a subset of the chosen data set (by letting the user restrict the data items included in the subset by selecting the range of each variable). Just like most interaction available in the SOM Visualizer, the selection of the range is done directly in the visualization component, in this case by dragging the minimum and maximum sliders to the positions desired.

The SOM algorithm can then be run on the specific region of the data set that appears to be of most interest. If the result did not match the users’ intentions, a new selection can easily be made and the algorithm can be run again. By only processing a subset of the data, the performance of the application can be greatly improved. Thus, this technique is especially effective when dealing with very large data sets.

When the SOM algorithm has been run, the user can choose to view which input data that corresponds to which cluster, i.e. perform drill-down on the SOM. This is done by simply clicking on a patch chart. The result is shown in a parallel coordinate plot.

Previously, SOM applications have only been dealing with the association of data items to the neurons. They have not been able to display which data items that have been grouped together. The interaction mentioned in the previous paragraph solves this problem too, by displaying the values of the original data items that have been associated with the selected neuron.

If several clusters are of interest, the user can choose to select these in the 3D surface map and view the corresponding input data. This new interactive three-dimensional visualization also makes it easier to perceive close groups of clusters. The possibility to rotate the view facilitates the understanding of how clusters are ordered on the map. The SOM Visualizer also gives the user a choice to hide or show the isolines in the 3D surface map by a simple click. By assigning a colormap to the surface map, a more intuitive visualization is achieved. The colormap together with the height differences in the map yields a faster interpretation.

The nature of the SOM algorithm yields a neuron grid in which neighboring neurons tend to have similar dimensional values. However, this does not automatically mean that the neighbors of a neuron are the neurons most similar to it. This and the fact that neuron grids having a large amount of neurons yield better results than those having only few neurons are the reasons why improvements of the neuron selection with a dynamic neighborhood have been made, i.e. the possibility to show the values of the most similar neurons as well as the neuron selected.
The development of interactive two and three-dimensional visualizations is an important extension of the ordinary static two- and three-dimensional views. Applications exist, which offer these visualizations, but none with the type of interaction the SOM Visualizer can provide. The interactivity combined with the numerous plots and charts displaying e.g. association frequencies, intra- and inter-neuron similarities and dimensional values, make the SOM Visualizer a useful tool for the analysis of any kind of multivariate data, especially if the analysis process focuses on similarity analysis.

The integration of the engine used in the SOM Visualizer in the Smartdoc scatter component is the core of the development of an algorithm for the selection of dimensions to display in a three-dimensional scatter plot. This is done by automated analysis of the output of the SOM algorithm to select three dimensions to represent the structure of the data set.
10 Assessment and Future Work

Discussions with the end users have been held continuously during the development process to keep focus on the functionality desired by the target group. In the final stage of the development process, the SOM Visualizer was thoroughly evaluated by the end users and by three research engineers at the department of science and technology at Linköping University.

When an early prototype was demonstrated for the end user, it was decided to keep a stronger focus on visualization and interaction and to cease the development of the SOM algorithm as it already fulfilled its purpose. The development of a three-dimensional SOM-grid was aborted, because it lacked the possibilities to create visualizations that were easy to interpret.

A request was made to create a way to translate the mapped data into the original space, and it resulted in the drill-down function used on the patch chart. Another request was to find a way to enhance the visualizations of the mapped data resulting in the inter- and intra-neuron relationship charts.

At the final stage of the development process, a beta version of the SOM Visualizer was evaluated by three research engineers in the area of information visualization. The evaluations included bug fixes and suggestions of new features. Some have been implemented while others had to be postponed due to the time limitations of the project.

Below is a summary of the suggestions for future work brought up at the meetings.

- If it was possible to mark a region of interest in the show neuron value window, and have the corresponding region of interest shown in the other charts, interpretation would be facilitated in some cases.

- It would be useful to improve the support of file formats, such as relational databases and regular text files.

- When huge data sets are analyzed, demands for improved performance could rise. It is possible to reduce calculation time and memory load by optimization of the code.

- Instead of the square grid currently used, a hexagonal grid would make interpretations of similarity more intuitive.

- Because the limitations in SOM regarding long distance neuron relationships, the addition of other similar algorithms would be useful. Some examples would be the RSOM and ViSOM algorithms, but other unsupervised learning algorithms would also be of interest.
• When working with large data sets, it would be useful to save various selections based on the SOM-grids. Another nice feature would be to save the SOM grids, as the calculation time of the SOM algorithm can be substantial in huge data sets.

• The shapes of the user defined selections could be made more flexible. For example, a useful feature would be to let the user draw a custom shaped selection.

• A more flexible way to select subsets of the original data would be to let the user select which dimensions that should be included in the calculations. The current application only supports restrictions in the ranges of the dimensional values.

• Support of dimensions based on non-unique strings. This could be done by first replacing the strings with numbers and then extracting each number to a new dimension by replacing the active number with a one and the inactive numbers with a zero for the active dimension. Consequently, the number of dimensions created by this process would equal the number of different strings in the original dimension. To restrict the importance of the newly created dimensions, the span of those dimensions should equal 1/n of those of the original dimensions where n is the number of different strings in the original dimension. The usage of this method would avoid the current limitations in string handling.

• The plot of the drill-down could be removed and instead be included as a selection in the input plot. This would reduce the size the program requires on the screen and make it possible to run SOM on selections made by drill-down in SOM-maps. The latter would be very interesting, especially when used with selection based on nearest neighbor calculations.
11 Conclusion

The assignment for this thesis consisted of two tasks; to create a stand-alone application used for analysis of large multidimensional data sets and to develop an algorithm based on a similarity criterion to select axis to visualize in a Smartdoc scatter component. The results of both tasks are presented below.

The SOM Visualizer is the result of a process aiming at developing a tool for analysis of data classified and ordered by a scaling algorithm. In this section the focus is set on how well the SOM Visualizer is fit to perform those tasks.

During the development of the application, about 80 percent of the time was spent on the main part of this task, the visualizations and the possibilities of user interaction in those visualizations. The rest of the time has been used on other areas such as the SOM algorithm and input of data. Consequently, the analysis should focus on the former.

The SOM Visualizer produces a variety of plots that each returns just a fraction of the information needed for full analysis of the data. Together though, the plots can be useful for anything from analysis of the characteristics of clusters of input data items to analysis of dimensional attributes and dependencies. The possibility to view the original data as well as the constructed cluster maps along with drill-down techniques, used to highlight the attributes of dynamically sized clusters in the original data set, make it a flexible tool for data analysis.

The flexibility is one of the major advantages with the application. It can take any properly structured Excel data set as input, select a desired subset and then run the SOM algorithm with a wide variety of options to return results that can be analyzed using several different tools including drill-down selection to produce dynamically sized selections in the original data set. Besides, the learning of the SOM algorithm is unsupervised so the user can concentrate on the data analysis.

The program creates both two- and three-dimensional visualisations depending on the user’s preference. A three dimensional neuron grid was also developed, but it was removed due to the interpretation problems it generated.

When it comes to ease of use, the program requires basic knowledge in neural networking. A user lacking such skill would probably face problems when it comes to the understanding of certain plots without studying the manual. User-friendliness has been one of the primary goals during the development process. For example, the colormaps are used to speed up understanding of similarities and differences and the analysis process is further simplified by the several intuitive ways the user can interact with the data directly in the charts. The fact that most of these interaction possibilities produce real-time results also improves the user-friendliness.

However, the application is far from optimized when it comes to speed and only one algorithm (SOM) is used to create the cluster maps. The program would benefit if
these issues and others mentioned in the section Future Work were taken care of, but there is no time at this stage.

The Smartdoc scatter component is used to select which dimensions to visualize when the number of dimensions in a data set exceeds the number of dimensions the output plot is able to handle. Whether it succeeds or not is in the eye of the beholder, as there is no such thing as a right or a wrong choice. The question must be how well the algorithm conforms to the criteria it is designed to follow.

The idea was to select three dimensions that are as different as possible. To some extent, that can be measured. By analysis of the data set, the choices of the component can be compared with what the human eye would choose.

To test the possibilities to analyse data sets using the SOM Visualizer and the Smartdoc scatter component, the data sets were visualized and three dimensions were selected. Then the Smartdoc scatter component was run and the same or to the human eye very similar dimensions were selected from the dimensions available as long as it was clear which dimensions to select manually. When it was harder to get an overview of the characteristics of the dimensions, a single manual choice was not easy to make, and consequently it was harder to evaluate the choices of the algorithm.

However, the test showed that the choices made by the algorithm were clearly better than the goal (to make better choices than random selection). More importantly, it also showed the usefulness of the SOM Visualizer.
12 ACKNOWLEDGEMENTS

This study is a thesis for the Master of Science degree in Media technology at the department of Science and Technology, Linköping University. It was accomplished from September 2002 to March 2003 in cooperation with Unilever and the EC founded project, Smartdoc IST-2000-28137.

We would like to give a special thanks to our examiner Professor Mikael Jern for giving us this assignment and for support, ideas and inspiration and to our assigner and advisor Dr. Robert Treloar at Unilever for ideas, great feedback and technical support.

Thanks to Staffan Palmberg, Magnus Ranlöf and Andreas Nilsson for helping us with the evaluation of the SOM Visualizer and for technical support.

We also want to thank Staffan Palmberg for helping us with the integration of the self-organizing map algorithm in Smartdoc and for letting us use the Excel-reader and the parallel coordinate plot component.
13 ABBREVIATIONS

GUI – GRAPHICAL USER INTERFACE
VUI – VISUAL USER INTERFACE
BASIC – BEGINNER’S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE
VBSCRIPT – VISUAL BASIC SCRIPTING EDITION
ASP – ACTIVE SERVER PAGES
IIS – INTERNET INFORMATION SERVER
VB – VISUAL BASIC
AVS – ADVANCED VISUAL SYSTEMS
COM – COMPONENT OBJECT MODEL
PCA – PRINCIPAL COMPONENT ANALYSIS
MDS – MULTIDIMENSIONAL SCALING
SOM – SELF-ORGANIZING MAP
ViSOM – VISUALIZATION INDUCED SELF-ORGANIZING MAP
HCI – HUMAN COMPUTER INTERACTION
STM – SHORT-TERM MEMORY
LTM – LONG-TERM MEMORY
## GLOSSARY

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract information</td>
<td>For example, graphs, multivariate data or multimedia objects.</td>
</tr>
<tr>
<td>Clustering</td>
<td>Reduces the amount of data by grouping similar data items together.</td>
</tr>
<tr>
<td>Competitive learning algorithms</td>
<td>Distribute a certain number of vectors in a possibly high-dimensional space.</td>
</tr>
<tr>
<td>Data element</td>
<td>One single point with coordinates $X_1, X_2, X_3, \ldots, X_n$ in the data set where $n$ is the number of dimensions.</td>
</tr>
<tr>
<td>Data Mining</td>
<td>The process of analyzing data from different perspectives and summarizing it into useful information.</td>
</tr>
<tr>
<td>Information Visualization</td>
<td>The process of forming a mental model of data, thereby supporting insight into that data.</td>
</tr>
<tr>
<td>Metrics</td>
<td>The way distance is measured.</td>
</tr>
<tr>
<td>Parallel Coordinates</td>
<td>Vertical axes each scaled to a different variable.</td>
</tr>
<tr>
<td>Neuron</td>
<td>A cluster containing similar data items</td>
</tr>
<tr>
<td>Scientific Visualization</td>
<td>A research area concerned with techniques that allow the creation of graphical representations from large data sets.</td>
</tr>
</tbody>
</table>
15 References


<table>
<thead>
<tr>
<th>Reference</th>
<th>URL</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
</table>
16 Appendix A

An example of how the smart filter algorithm works.

1. Run the SOM algorithm on the multidimensional dataset. This returns a two-dimensional grid of neurons.
   The example is based on a grid of 3x3 neurons.
   
   Neuron (1 1) has the weight vector (1,4,6,8,9,100)
   Neuron (1 3) has the weight vector (1,2,5,2,9,100)
   Neuron (2 3) has the weight vector (5,8,2,1,1,900)

   Do this for all other combinations accordingly.

2. Normalize the weight vector values.
   Assume all the dimensions have a range of 0-9 except dimension 6 that has a range of 0-999. The normalized vectors would be:
   
   Neuron (1 1) has the weight vector (1,4,6,8,9,1)
   Neuron (1 3) has the weight vector (1,2,5,2,9,1)
   Neuron (2 3) has the weight vector (5,8,2,1,1,9)

3. Calculate the differences between the (normalized) values in different dimensions. Because only three neurons of the 9 are mentioned the calculations below are made assuming these three are the only ones.

   Ex. diff(1,2) means the difference between the first and the second dimension (the sum for all neurons)
   \[ \text{diff}(1,2) = |1-4| + |1-2| + |5-8| + [\text{all diffs for all neurons not mentioned}] = 7 [\text{+all diffs}] \]
   \[ \text{diff}(1,3) = |1-6| + |1-5| + |5-2| = 12 \]
   \[ \text{diff}(1,4) = |1-8| + |1-2| + |5-1| = 12 \]
   \[ \text{diff}(5,6) = 8 + 8 + 8 = 24 \]

   For practical reasons some more are made up:
   \[ \text{diff} 1,5 = 8 \]
   \[ \text{diff} 1,6 = 2 \]
   \[ \text{diff} 3,4 = 5 \]

   The program would calculate the differences between all dimensions possible, but in this example, the values above are sufficient

4. Sort the differences (ascending)
   \[ \text{diff} 1,6 = 2 \]
   \[ \text{diff} 3,4 = 5 \]
   \[ \text{diff} 1,2 = 7 \]
   \[ \text{diff} 1,5 = 8 \]
   
   The results are stored in an array called sortedDiffs
5. Now the grouping starts.
The grouping begins with n groups where n is the number of dimensions. In this case six (initially the groups consist of one dimension each).
Then, the number of groups is reduced, one at a time until only three groups are left.

First, create a new group consisting of the most similar dimensions. This is done as follows:

A. Cluster the most similar dimensions, in this case 1 and 6 (now there are only five groups)

B. Then, cluster the second most similar dimensions, in this case 3 and 4 (the five groups have become four)

C. Next, cluster the dimensions third in the sortedDiffs array (1 and 2). Since 1 already belongs to a group, make 1 the representative dimension for that group. Also, from now on, all differences in the sortedDiffs array including the passive dimensions (2 and 6) are ignored.

D. Continue clustering like this until only three groups are left. In the example, this happened already in step three (since there are only six dimensions in this example). The groups are:

- 1, 6, 2
  In this group 1 is the representative and consequently one of the three dimensions to show in the scatter plot.

- 3, 4
  Since this group consists of only two dimensions, no representative was chosen in the last step. Either dimension could be chosen as the representative, let's pick the first one (3).

- 5
  The last group consists of dimension 5 only.

6. Now the three dimensions are chosen. Dimension 1, 3 and 5 will be used in the scatter plot.

Note:

- If two groups are to be clustered, two alternative proposals are presented:

The first is: do not cluster groups. Ignore that row in sortedDiffs and look at the next one. Since this might lead to that the number of groups exceeds 3 when all the values the sortedDiffs list have been processed, some kind of selection has to be made. One option is that the biggest groups are selected first. If two groups are of equal size and
only one could be included among the three dimensions, choose the one with the smallest sum of differences.

The second is: cluster groups and let the representative of the larger group become the representative.

If 1, 6, 2 are to be clustered with 3, 4 then make a group consisting of 1, 2, 4, 6 and make 1 the active dimension because it was representative for the larger group.

The description above is based on two assumptions:

The dimensions selected do not have to be the “best” or “optimal”. Not all alternatives are checked. This is illustrated by an example: If dim 1 and 2 are similar, 2 and 3 are similar and 3 and 4 are similar, then 2 would become the representative (for a group consisting of 1, 2, 3, 4) if the values are grouped in that order. But 2 and 4 might be very different nonetheless.

The other assumption is that “good” picks for the selection of 3 dimensions are 3 dimensions that are as unlike each other as possible, since the assumption is made that similar dimensions can be approximated as “about the same” as dimension x in the scatter plot.
User Manual
Introduction
This manual should be read carefully in order to fully understand all features and limitations of the SOM Visualizer. This manual assumes that the user have basic computer skills and basic knowledge in information visualization. Words written in **Bold** represent actions from the user, i.e. clicking on a button etc. Words in *Italic* represent available parameter options.

The description below is for the Basic mode with the default settings, which is recommended for novice users. For a full description of parameter settings and options, see the advanced mode section. Figure 1 shows a simplified layout of the application that facilitates the orientation. The image is referred to later in the manual.

Why Use the SOM Visualizer?
The SOM Visualizer is an analytical tool used to speed up the finding of trends and anomalies in large multidimensional data sets. It produces several different visualizations based on the SOM algorithm, that combined simplify the analysis process and make decision-making more accurate.

Requirements
The SOM Visualizer can be used on any computer with the operation system Windows NT/2000/XP. The program requires Microsoft Visual Basic 6 runtime library *(VBRun60sp4.exe*, which can be found at: [http://www.simtel.net/vbrun.php](http://www.simtel.net/vbrun.php)), AVS Oopencviz library *(ov22_viewer_com.exe*, which can be found at: [http://servus.itn.liu.se/Smartdoc/*](http://servus.itn.liu.se/Smartdoc/*)) and Microsoft Office.

There are no known limitations regarding RAM or graphic cards, but when dealing with large data sets, it is preferable to have 128 MB RAM and a graphics card with a minimum of 64 MB.

Starting Up
When the SOM Visualizer starts, the first thing to do is to load a data set. This is done by clicking on the **Browse** button in the upper left corner of the program. A window appears that enables browsing for the file. The program supports the MS Excel file format .xls. When the file is selected click **Open** and the file is loaded into the program.
The SOM Visualizer then filters the metadata from the data containing the multidimensional data. Then the data is visualized in a parallel coordinate plot, view A in figure 1.

**Running the SOM-Algorithm**

When a file has been loaded, the SOM algorithm can be run. If only a selection of the data set is of interest, a selection can be made by using the two different types of arrows in figure 2. However, this is optional as all data is selected by default.

![Figure 2 - Arrows used for selection](image)

Clicking on the **Run SOM** button executes the algorithm and the application starts assigning the data items to the cluster grid. The menu bar to the left shows some information about the size of the data set and the calculation progress.

When the SOM algorithm has been run, the patch chart in view C is displayed showing a two dimensional grid of neurons. The brighter a neuron is, the more input vectors are associated with it. The visualization in view D shows the standard deviation in every cluster. This second patch chart is a complement to the chart previously mentioned. It displays the intra-neuron differences, their values spanning from blue (small relative difference) over green and yellow to red (big relative difference).

Clicking on the **Show Neuron Values** button opens a new window. This window displays the neuron value for each dimension and gives a good overview of the data set. These plots can be very helpful when it comes to the detection of trends and anomalies in the data.

Clicking on the radio buttons in the menu bar (see figure 3) switches between the three-dimensional surface map and the two-dimensional patch chart.

![Figure 3 - Switching between visualizations](image)
Interactivity
The SOM Visualizer offers plenty of interactivity. By clicking on the patch chart visualization in view C or D, the original data associated with each clusters and the five most similar neurons are shown in the lower parallel coordinate (view B). In the surface map a rectangle is used for selection. By clicking with the mouse pointer, the size of the rectangle can be changed and the result of the selection is shown in the parallel coordinate plot in view B.

Advanced Mode
Clicking the Advanced Mode button switches the SOM Visualizer to advanced mode. This mode is recommended for advanced users and gives the possibility to change the SOM settings, the parameters for the surface map and the Patch chart.

SOM Settings
Figure 4 shows the SOM settings which are used to change the parameters of the SOM algorithm.

![SOM Settings](image)

**Figure 4 - SOM settings**

**Metric Function**
This setting decides the metric to use when the winner is determined. The options are City-block, Euclidean (default) and 8-connectivity.

\[
\text{Cityblock} = \Delta X + \Delta Y
\]

\[
\text{Euclidean} = \sqrt{X^2 + Y^2}
\]

8-connectivity = \max(\Delta X + \Delta Y)

**Neighborhood**
This setting decides how the neighborhood distance and the neighborhood radius are combined to alter the way the distance from the winning neuron affects the learning. The options are Triangular (default), Gaussian, and Excitation & Inhibition. A triangular neighborhood simply lets the learning rate within the allowed neighborhood radius decrease linearly with the distance to the winning neuron. A Gaussian
neighborhood has the form of a Gaussian curve as the learning decreases exponentially. A neighborhood based on Excitation & Inhibition is calculated as follows:

\[ 1 - \left( \frac{N_d}{N_r} \right)^2 \times \exp \left( -\frac{N_d}{N_r} \right)^2 \]

\[ N_d = \text{Neighborhood Distance} \]
\[ N_r = \text{Neighborhood Radius} \]

**Neighbor Metric**

Neighbor Metric decides the metric to use when the neighborhood is calculated. It uses the same options as the metric function.

**Learning Rate**

Each time an input vector has been associated to a neuron, the neighboring neurons are updated according to their distance to the winning neuron based on the neighborhood setting and the neighborhood radius. However, the number of times the neurons have won also affect the size of their learning factors. That is exactly what the learning rate does. The options are: Constant, Linear Decay (default), Square Root Decay and Exponential Decay.

**Initial Learning Rate**

The initial learning rate is the base on which the learning rate calculation is applied. A value around the default value (0.55) seems to minimize the error measurement calculations included in the program for most data sets and settings.

**Neurons per Dimension**

The size of the neuron grid is one of the options that have the biggest impact on calculation time. The fact that it also has a major impact on the result makes it the most obvious factor when it comes to the trade-off between speed and accuracy. For data sets containing less than 1000 items, neuron grids with 20 or less neurons per dimension have near real-time calculation times.

**Neighborhood Radius**

The neighborhood radius regulates the number of neighbors that get their positions updated because of the assigning of an input vector.

**Pre-process Neuron Values**

Normally, the algorithm assigns the initial values of the neuron grid randomly. When pre-processing of the neuron values is checked, the algorithm makes qualified guesses about what is a good neuron grid. The original algorithm does the following:

- Assigns the neuron grid randomly
- Runs SOM and adjusts the neuron grid to the input vectors simultaneously

When Pre-process Neuron Values is selected, the procedure is:

- Assign the neuron grid
• Simulate a SOM run, but use the winners only to adjust the neuron grid
• Run SOM and adjust the neuron grid further

The pre-processing improves the accuracy with up to 40%. Despite that, pre-processing is turned off by default to decrease computation time.

**Patch Chart**
Under the patch chart advanced settings (see figure 5) the number of similar clusters shown can be changed. The checkbox is checked as default and shows the five closest clusters. If the checkbox is left unchecked only the original data in the current cluster is shown in view B. The default patch chart shows the *IntraNeuron Standard Deviation*, which is a measurement of how similar the clusters are among themselves. If the *Neighbor Difference* is checked, the patch chart shows how similar the clusters are to their neighbors.

![Figure 5 - Patch chart advanced option](image)

**Surface Map**
With the advanced option for the surface map, the isolines and the crop rectangle can be turned on and off. This is done by clicking on the *Show Isolines* and *Show Crop Rectangle*, see figure 6. The surface map can also be used to visualize the neighbor difference and the intra-neuron standard deviation in three-dimensions, see figure 6.

![Figure 6 - Surface map advanced option](image)

The surface map has a default color map from blue, which represent low values, to red, which represent high values. In the advanced mode the color map can be specified by the user. To change the different minimum and maximum values, click on the blue or
the red area, see figure 7. This displays a color palette, which makes it easy to change colors.

Figure 7 - Colormap