Information Visualization with Self-Organizing Maps

Jing Li

Abstract: The Self-Organizing Map (SOM) is an unsupervised neural network algorithm that projects high-dimensional data onto a two-dimensional map. The projection preserves the topology of the data so that similar data items will be mapped to nearby locations on the map. Despite the popular use of the algorithm for clustering and information visualisation, a system has been lacking that combines the fast execution of the algorithm with powerful visualisation of the maps and effective tools for their interactive analysis.
Powerful methods for interactive exploration and search from collections of free-form textual documents are needed to manage the ever-increasing flood of digital information. In this article we present a method, SOM, for automatic organization of full-text document collections using the self-organizing map (SOM) algorithm. The document collection is ordered onto a map in an unsupervised manner utilizing statistical information of short word contexts. The resulting ordered map where similar documents lie near each other thus presents a general view of the document space. With the aid of a suitable (SVG) interface, documents in interesting areas of the map can be browsed.

1 Introduction

Over the past decade, academic as well as commercial informations have been growing at exceptional rates. Gaining new knowledge from such databases is difficult, costly and time-consuming if done manually. It may even be impossible when the data exceeds certain limits of size and complexity. As a result, the automated analysis and visualisation of massive multi-dimensional datasets has been the focus of much scientific research during the last years. The principal objective is to find regularities and relationships in the data, thereby gaining access to hidden and potentially useful knowledge.

Artificial Neural Networks are a promising part of this broad field. Inspired by advances in biomedical research, they form a class of algorithms that aim to simulate the neural structures of the brain. The Self-Organizing Map (SOM) is a fairly well-known neural network and indeed one of the most popular unsupervised learning algorithms. Since its invention by Finnish Professor Teuvo Kohonen in the early 1980s, more than 4000 research articles have been published on the algorithm, its visualization and applications. The maps comprehensively visualise natural groupings and relationships in the data and have been successfully applied in a broad spectrum of research areas ranging from speech recognition to financial analysis.

The Self-Organizing Map performs a non-linear projection of multidimensional data onto a two-dimensional display. The mapping is topology-preserving, meaning that the more alike two data samples are in the input space, the closer they will appear together on the final map. This allows the user to identify ‘clusters’, i.e. large groupings of a certain type of input pattern. Further examination may then reveal what features the members of a cluster have in common.

2 Background

Data Mining is a broad area of research concerning the "extraction of implicit, previously unknown and potentially useful information from data" [Witten and Frank (1999)]. Increasingly powerful computer hardware has made it possible to explore and investigate databases whose complexity, dimensionality and amount of data exceed the limits in which manual analysis is possible. The purpose is to reveal patterns, relationships or regularities that allow us to gain new knowledge and insight on the data.

Machine Learning provides the technical basis for Data Mining and the Self-Organizing Map (SOM) is one of the many algorithms used in this context. More precisely, the SOM belongs to the class of Neural Network algorithms. This is a group of algorithms based on analogies to the neural structures of the brain. The SOM in particular was inspired by an interesting phenomenon: As physicians have discovered, some areas of brain tissue can be ordered according to an input signal [Kohonen (1982)]. Basically, the SOM is a computer program simulating this biological ordering process. Applied to electronic datasets, the algorithm is capable of producing a map that shows similar input data items appearing close to each other. An example of this is shown in Figure 1 below, taken from [Vesanto (1999)].

Figure 1: An example of a trained map showing groups of similar data items

There are numerous applications involving the SOM algorithm but the most widespread use is the identification and visualisation of natural groupings in the data. This process of finding similar items is generally referred to as *clustering*.

2.1 Artificial Neural Networks

Advances in biological research have offered an initial understanding of the natural thinking mechanisms that make it possible for humans to learn from previous experience. It appears that the fundamental processing elements of the brain are the neurons, a vast number of special, interconnected cells. These cells communicate via electrochemical pathways and allow the brain to store information as patterns.

Inspired by the examinations of the structures of the brain, Artificial Neural Networks are computer simulations of 'brain-like' systems of interconnected processing units. The processing units are often referred to as nodes and typically viewed as being analogous to the neurons of the human brain. Real neural networks - in contrast to their artificial counterparts - are not binary, not synchronous and not stable systems. But the very basic behaviour of the artificial processing units is similar to the one of the biological neurons: First, the unit receives signals from a number of other nodes in the network. The sum of these signals determines the unit's internal level of activity. Secondly, the combined signal is passed on to other nodes through a weighted connection.

The pattern of connectivity of an Artificial Neural Network (i.e. The combination of weights associated with each connection) determines the system's response to an input stimulus. Given a certain pattern of weights, an input x leads to a predictable output y. In this sense, the artificial neural network acts similar to a conventional computer program. In strong contrast to the conventional program, however, the neural network does not require a step by step procedure to perform a desired task. Instead, the network can be taught to do the task. This is the so-called training process.

At the beginning of the training process, the weights associated with the node connections will typically be set to random values. This corresponds to the network knowing nothing. As the training process proceeds, the weights will converge to values allowing them to perform useful computation. We can thus say that the network commences knowing nothing and moves on to gain some real knowledge. This is referred to as learning.

The learning process can be competitive, meaning that during each training step the particular node is determined that is already closest to the input signal. The node is rewarded by being allowed to adapt even further to the input - in other words: to learn more. If the map receives feedback from the database or outside intervention from the user is necessary, the learning phase is said to be supervised. It is called unsupervised, if the algorithm learns about the data merely by inspecting it.

2.2 The Self-Organizing Map

2.2.1 Introduction

The Self-Organizing Map belongs to the class of unsupervised and competitive learning algorithms. It is a sheet-like neural network, with nodes arranged as a regular, usually two-dimensional grid. As explained in the previous section on Neural Networks, we usually think of the node connections as being associated with a vector of weights. In the case of Self-Organizing Maps, it is easier to think of each node as being directly associated with a weight vector.

See Figure 2 below for an illustrative representation of a 4 by 3 map with 3-dimensional weight vectors.

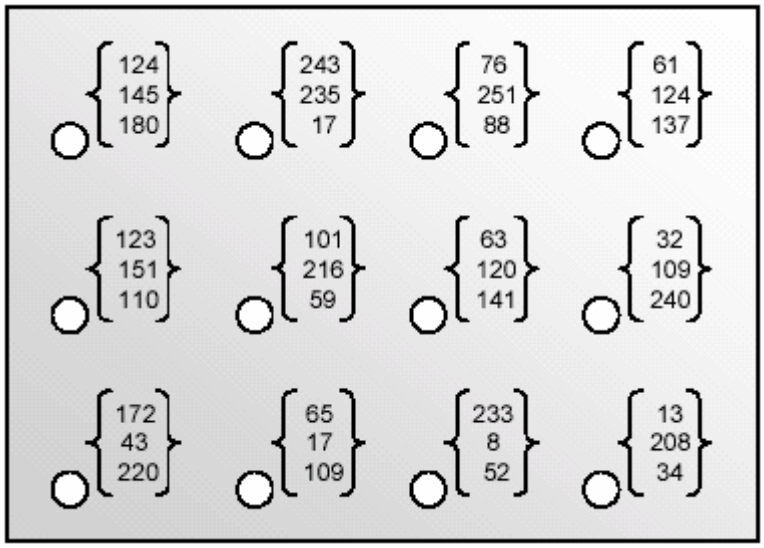


Figure 2: Each map node is associated with a vector of weights

The items in the input data set are assumed to be in a vector format. If n is the dimension of the input space, then every node on the map grid holds an n-dimensional vector of weights.

mi =[mi1 , mi2 , mi3 , . , min] (Equation 2)

The basic principle of the Self-Organizing Map is to adjust these weight vectors until the map represents .a picture. of the input data set. Since the number of map nodes is significantly smaller than the number of items in the dataset, it is needless to say that it is impossible to represent every input item from the data space on the map. Rather, the objective is to achieve a configuration in which the distribution of the data is reflected and the most important metric relationships are preserved. In particular, we are interested in obtaining a correlation between the similarity of items in the dataset and the distance of their most alike representatives on the map. In other words, items that are similar in the input space should map to nearby nodes on the grid.

2.2.2 The Algorithm

The algorithm proceeds iteratively. On each training step a data sample x from the input space is selected. The learning process is competitive, meaning that we determine a winning unit c on the map whose weight vector mc is most similar to the input sample x.

$$||x-w_c||=\min_I||x-w_I||$$

The weight vector mc of the best matching unit is modified to match the sample x even closer. As an extension to standard competitive learning, the nodes surrounding the best matching unit are adapted as well. Their weight vectors mi are also .moved towards. the sample x. The update rule may be formulated as:

$$\mathbf{w}j(t+1) = \mathbf{w}j(t) + m(t)\, lw(\mathbf{x})(j,t)\, [\mathbf{x} - \mathbf{w}j(t)]$$

where $lw(\mathbf{x})(j,t)$ is the ‘degree of neighbourhood’ of node j w.r.t. the winning node $w(\mathbf{x})$ at time t, m(t) is the learning rate at time t,

the term in square brackets represents the ‘difference’ between the input vector and the weight vector (usually Euclidean distance).

2.2.3 The choice of parameters

The maps produced with the SOM algorithm are very much influenced by our choice of parameters. This includes

- the map width and height,
- the number of iterations,
- the size the initial radius and
- the initial value of the learning rate.

The are no strict guidelines for choosing any of these parameters. A process of .trial and error. is necessary to determine a set of values that are suitable for the dataset at hand. As a rule of thumb, it has been suggested [Kohonen et al (1996)] to use rectangular (but non-quadratic) maps, say, of size 15 by 10 and to use an initial radius equal to the height of the map. For the value of the initial learning rate factor a value of 0.05 has been suggested.

In the following diagrams, we have assumed

- Total number of iterations: 10000
- Initial Radius: 10
- Initial Learning Rate: 0.05

Figure 3 shows how the learning rate and the radius decrease linearly as the number of cycles increases. The learning rate goes down to 0 but the radius is at least one.

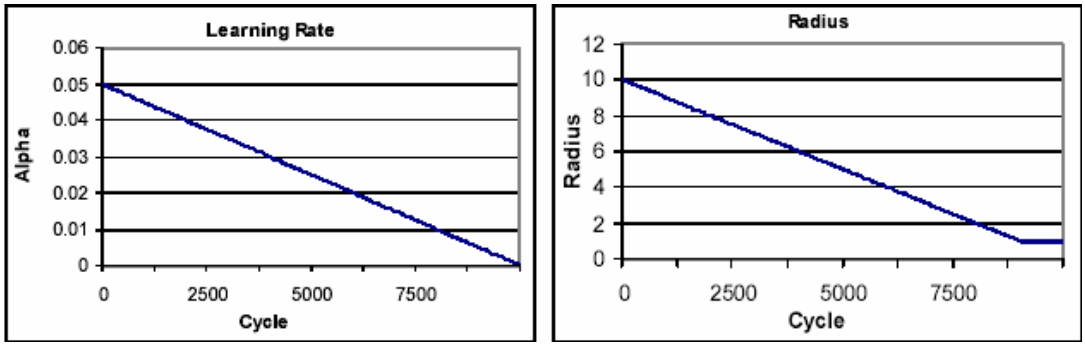


Figure 3: Learning Rate and Radius decrease monotonically

2.2.4 Initialization of the map

Three different techniques are commonly used to initialize the map. The first approach is to use random values, completely independent of the training data set. This corresponds to the map knowing nothing about the input data. It is a simple but rather poor way of initializing the map because it requires quite a number of additional training cycles until the map can be said to be at least roughly representative of the training data.

The second approach is to use random samples from the input training data. The advantage is that the initial weight vectors already lie in the same space as the training data. When the training commences, the map is already in a state in which it represents at least a subset of the input data items. This obviously reduces the number of training iterations needed and hence lowers the computational cost. Nevertheless, the choice of input samples used for the initialization is random and the number of map nodes is very small compared to the number of training data items. The initial map is hence not likely to be truly representative of the given dataset. One may easily imagine a scenario where by pure coincidence a large number of outliers have been chosen which have little in common with the majority of data items. Similarly, the initial samples may not reflect the existence of outliers at all.

The third approach to map initialization tries to reflect the distribution of the data more faithfully. This is not easy to achieve since the map is usually only two-dimensional while the dataset is often of a much higher dimensionality. One may consider in abstract terms how a multidimensional dataset can be best characterized using a reduced number of dimensions. Supposing we were to chose only one dimension to describe the dataset, which one should we chose? Clearly, if there is one dimension which always holds values very close to its mean, then it would be relatively easy to predict the values of this dimension. In other words, the information content of this dimension would be low. On the contrary, the dimension showing the strongest variation is of the greatest interest.

2.2.5 Available software implementations

Despite the popularity of the algorithm, relatively little work has been carried out in terms of software implementations. Apart from the many tools with purely educational purposes there are really only two software packages available that achieve a professional standard and are accepted in the academic community. Both of these have been published by the work group around SOM-inventor Teuvo Kohonen at the University of Helsinki, Finland.

The first one of the two packages is SOM_PAK [Kohonen (1996)] which dates back to 1996. The software is a text-only implementation written in ANSI C. It runs very fast but is inconvenient to

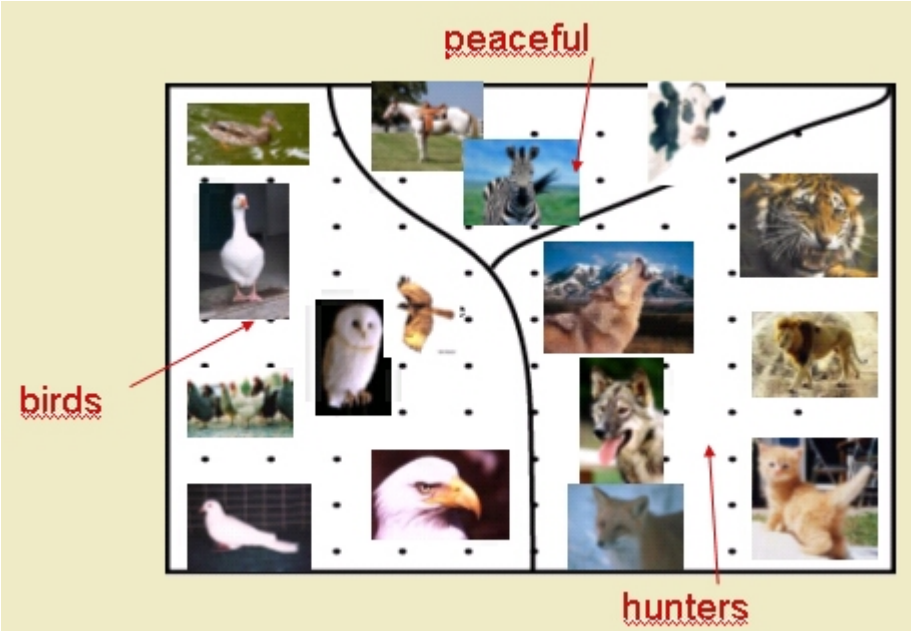
handle and includes hardly any visualisation support.

The other package is the SOM Toolbox [Vesanto et al (1999)]. The Toolbox is a software library for Matlab. It is strong in terms of map visualisation but performs rather poorly due to its integration into the Matlab environment. According to the authors of the Toolbox, release 1 of the software required 23 times as much time to perform the same map training task as the SOM_PAK tool. It also misses effective features for the analysis of the maps and cannot visualise an ongoing training process. The Toolbox provides some graphical user interface components but their handling is cumbersome and the use of the command line functions is strongly recommended by the authors of the Toolbox. Continued work on the software has led to a second release in 1999. By 2002, however, most of the departmental staff that has been involved with the Toolbox has left the University of Helsinki so further development of the package is uncertain.

2.2.6 Example of self-organizing maps

To illustrate the self-organizing result with a concrete model simulation, consider the data given in Table. Each column is a schematic description of an animal, based on the Presence (=1) or absence (=0) of some of the 13 different attributes given on the left. Some attributes, such as “feathers” and “2 legs” and correlated, indicating more significant differences than the other attributes, but we shall not take this correlation into account in learning in any way. In the following, we will take each column for the input vector of the animal indicated at the top .The animal name itself does not belong to the vector but instead specifies the label of the animal in the calibration of the map.
The members of the data set were presented iteratively and in a random order to a SOM of 10*10 neurons subject to the adaptation process described above. The initial connection strengths between the neurons and their n=29 input lines were chosen to be small random values, i.e. No prior order was imposed. However, after a total of 2000 presentations, each neuron became more or less responsive to one of the occurring attribute combinations and simultaneously to one of the 16 animal names, too. Thus we obtain the map shown in Figure 4.It is very apparent that the spatial order of the response has captured the essential “family relationships” among the animals. Cells responding to, e.g. “birds” occupy the left part of the lattice, “hunters” such as “tiger”, ”lion” and “cat” are clustered toward the right, and more “peaceful” species such as “zebra” , “horse” and “cow” are situated in the upper middle. Within each cluster, a further grouping according to similarity is discernible.

		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
is	Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0



2.2.7 Visualizing Self-Organizing Maps

Extracting the visual information provided by the Self Organizing Map is a central concept of this paper. The choice of visualization technique, however, is far from straightforward.

Visualizing a SOM is challenging because the input data is usually of a high dimensionality. By projecting the input space to a two-dimensional grid we can express the similarity of two samples as the distance between them. But while simplicity is gained by reducing dimensionality, information is effectively lost when the data item is simply represented by a dot. The mere position on the map cannot sufficiently embody the complexity of an n-dimensional vector.

The problem of visualising multivariate data is, of course, not a new one. Information Representation is a mature area of research and numerous approaches of displaying multidimensional multivariate data have been proposed. [Wong and Bergeron (1997)] The following paragraphs briefly review a number of these methods.

One of the simplest ways of coping with higher dimensions, is to use a two-dimensional coordinate plane and to incorporate further axes for each additional dimension, as shown in Figure 5. A vector can then be plotted as a curve connecting the axes. The intercept at axis i corresponds to the value of the multidimensional object in dimension i.

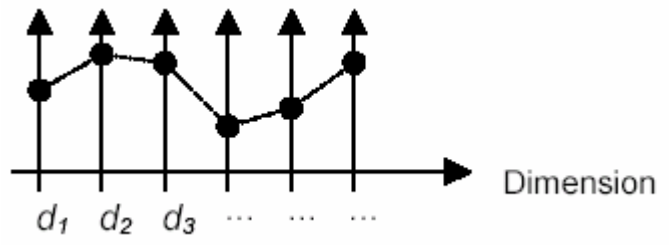


Figure 5: Visualising high-dimensional data using parallel coordinates

The parallel coordinates method has two advantages: Firstly, it is very useful in determining relationships between dimensions, say, of the form "Whenever d_2 is high, then d_4 is low". Secondly, the method scales well (i.e. linearly) with the number of dimensions, since we only need to add a further axis for each additional dimension. It is also possible to display a large number of objects in a single parallel coordinate's diagram. It is however not ideally suited for displaying a large number of objects in separate diagrams. So for Self-Organizing Maps, a more compact representation is favourable that allows us to plot a diagram for each node on the map. [Honkela (1997)] has suggested an iconified version which is shown in Figure 6.

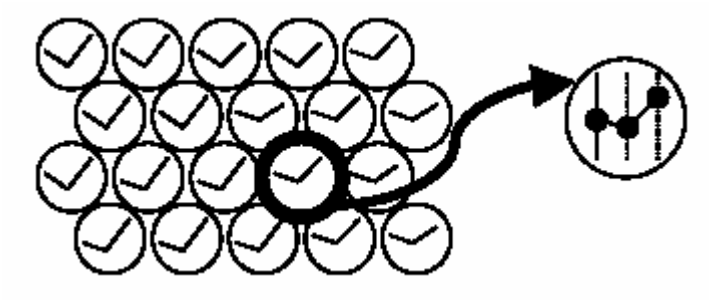


Figure 6: Honkela.s suggestion for visualizing multidimensional data

A large number of visualization techniques map each dimension of the input space onto a certain feature of the icon. This idea is perhaps best illustrated by introducing Chernoff.s Faces [Chernoff (1973)]. This rather peculiar way of graphically displaying multidimensional uses simple, cartoon-like faces (see Figure 7). Each dimension of the input vectors is assigned to a facial characteristic, e.g. nose size, eye spacing, mouth width, etc. The method draws upon the human ability to recognize small differences in facial characteristics and to digest many of these characteristics at once.

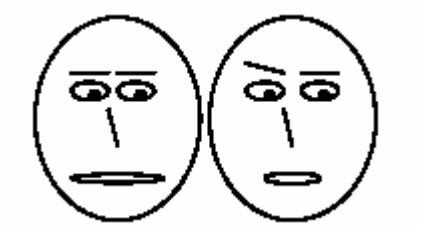


Figure 7: Chernoff.s faces

2.3 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is an XML markup language for describing two-dimensional vector graphics, both static and animated. It is an open standard created by the World Wide Web Consortium, which is also responsible for standards like HTML and XHTML.

SVG allows three types of graphic objects:

- vector graphic shapes (e.g. paths consisting of straight lines and curves, and areas bounded by them)
- raster graphics images / digital images

♦ text

```
<svg width="150" height="150" version="1.1" >

    <line x1="50" x2="140" y1="10" y2="40"
        stroke="green" stroke-width="2" />
    <circle cx="30" cy="30" r="10" fill="none"
        stroke="red" stroke-width="2"/>
    <rect x="50" y="50" width="40" height="20" stroke="black"/>
    <ellipse cx="120" cy="90" rx="25" ry="10"
        stroke="black" stroke-width="2" fill="darkgrey"/>
    <text x="20" y="90" font-size="10" fill="orange">
        SVG
    </text>
    <path d="M28,120 C28,120 38,96 64,117 C90,137 103,120 104,120"
        stroke="rgb(26,145,182)" fill="rgb(26,145,182)" stroke-width="1" />
</svg>
```

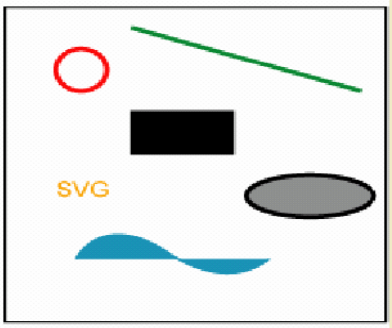


Figure 8: Example of SVG

Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. Text can be in any XML namespace suitable to the application, which enhances searchability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.

SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows straightforward and efficient vector graphics animation via ECMAScript or SMIL. A rich set of event handlers such as onmouseover and onclick can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same web page.

3 Software model for Information Visualization of SOM

The objective is to build a generic system based on the Self-Organizing Map algorithm. The system must be capable of generating and visualising maps, thus displaying the cluster relationships hidden in the data.

The task can roughly be broken down into the following steps:

- reading in a training data file

- initialising a new map or loading an old one

- training the map by running the algorithm

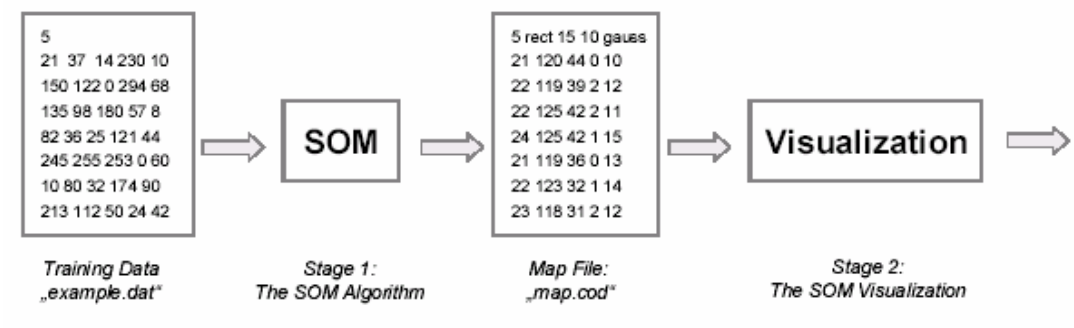
- writing the results to a file

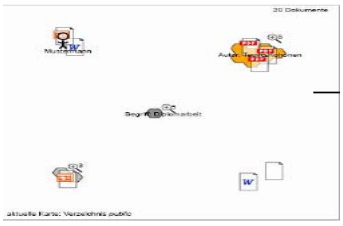
- translating the map into a graphical display such that the cluster structure of the data becomes visible and the relationships between the different dimensions are made explicit

- enabling the user to retrieve information on selected map nodes or regions

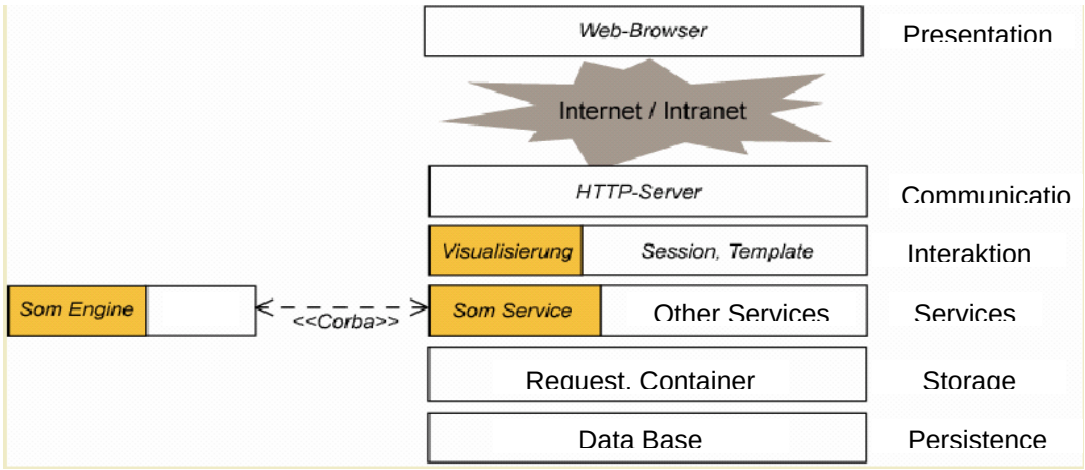
- offering the user to trace which input items correspond to a selected map region and enabling him to save these to a file for further processing

It is desirable to distinguish the algorithm from the visualization as clearly as possible. The anticipated System Structure is shown below.





The overall System architecture is shown below.



4 Conclusion

SOM is a highly useful multivariate visualization method that allows the multidimensional data to be displayed as a 2-dimensional map. This is the main advantage of SOM. The map units clustering makes it easy to observe similarities in the data. Through our experiment, we demonstrated that the possibility of quick observation of relationship between component (feature) and the class as well as the relationship among different component (feature) of the dataset from the visualization of a dataset. SOM is also capable of handling several types of classification problems while providing a useful, interactive, and intelligible summary of the data.

However, SOM also has some disadvantages. For example, adjacent map units point to adjacent input data vector, so sometimes distortions are possible because high dimensional topography can not always be represented in 2D. To avoid such phenomenon, training rate and the neighborhood radius should not be reduced too quickly. Hence, SOM usually need many iterations of training. And SOM also does not provide an estimation of such map distortion.

Alternatives to the SOM have been developed in order to overcome the theoretical problems and to enable probabilistic analysis. Examples of these include the Generative Topographic Mapping and the approach taken by Utsugi. Both of these approaches explicitly include a generative density model, which is a constrained Gaussian mixture model. Model selection can then be based on maximum likelihood or Bayesian evidence or some other well defined criterion.

Nevertheless, SOM still have many practical applications in pattern recognition, speech analysis, industrial and medical diagnostics, data mining.

5 Literaturverzeichnis

[Witten and Frank (1999)] Witten, I.H. and Frank, Eibe. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, San Francisco, CA, USA. 1999

[Kohonen (1982)] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. Biol. Cybernetics, volume 43, 59-62

[Kohonen (1995)] Teuvo Kohonen. Self-Organizing Maps. Springer, Berlin, Germany

[Vesanto (1999)] SOM-Based Data Visualization Methods, Intelligent Data Analysis, 3:111-26

[Kohonen et al (1996)] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, "SOM PAK: The Self-Organizing Map program package, " Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Jan. 1996

[Vesanto et al (1999)] J. Vesanto, J. Himberg, E. Alhoniemi, J Parhankangas. Self-Organizing Map in Matlab: the SOM Toolbox. In Proceedings of the Matlab DSP Conference 1999, Espoo, Finland, pp. 35-40, 1999.

[Wong and Bergeron (1997)] Pak Chung Wong and R. Daniel Bergeron. 30 Years of Multidimensional Multivariate Visualization. In Gregory M. Nielson, Hans Hagan, and Heinrich Muller, editors, Scientific Visualization - Overviews, Methodologies and Techniques, pages 3-33, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[Honkela (1997)] T. Honkela, Self-Organizing Maps in Natural Language Processing, PhD Thesis, Helsinki, University of Technology, Espoo, Finland

[SVG wiki] http://en.wikipedia.org/wiki/Scalable_Vector_Graphics

[Jost Schatzmann (2003)] Final Year Individual Project Report Using Self-Organizing Maps to Visualize Clusters and Trends in Multidimensional Datasets Imperial college London 19 June 2003