The Neural Data Analysis environment

Erkki Häkkinen Pasi Koikkalainen

University of Jyväskylä
Department of Mathematics
P.O.Box 35, FIN-40351 Jyväskylä, Finland

Abstract

From our viewpoint data analysis is a human controlled process. It is an iterative and interactive task, where the initial guess of data behavior may have to be changed several times. Thus, the effectiveness of analysis depends on what can be achieved during one iteration, and how long it takes to do it.

One part of the analysis is to reduce the amount of information such that its essential elements are maintained, and another is to display its contents in human understandable form. Our solution is to use variants of the self-organizing map (SOM) as an analysis method. This is a quantitative approach that can be adapted also as a part of a qualitative analysis.

1 Introduction

What are the requirements to which the Neural Data Analysis environment (NDA) should be an answer to ? Our approach follows the general idea of qualitative data analysis [5], as presented in figure 1. It has four abstract parts: data collection, data reduction, visualization, and conclusion making. In our case the SOM works as a reduction method, and the visualization is strongly supported by the structure of the neural network. Conclusion making is human responsibility, indicating that we consider the user and his controls as an integral part of the process. This choice puts a lot of emphasis on interface design. Especially data collection, data manipulation, data selection, and the making of visual presentations should feel as easy as possible.

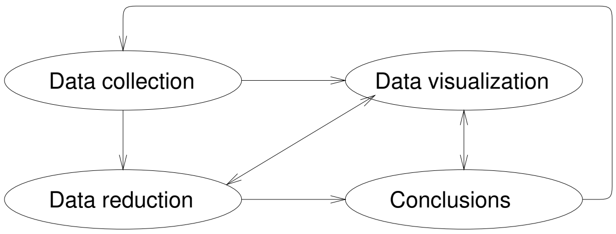

Figure 1: The abstract model of data analysis.

1.1 Why do we need iterative processing ?

In the case of unfamiliar data, an analyst can often define the final goals (e.g. find good customers), but the relation between the goals and input data (e.g. purchases and cost data) is often too superficial to build a direct mapping via any reduction method. Final goals can be impossible to achieve by one iteration, and the global problem might have to be divided into several subtasks.

Also, it often turns impossible to define beforehand what these subgoals are. Therefore, an iterative analysis is needed to find right questions and right answers for them.
As a result, we see the conclusion making as a very important part of analysis since only the human analyst can solve whether or not an operation fails to give an answer to the question. To illustrate, assuming that a method has been selected correctly and data includes all the information for the problem, the following examples show how one can fail in an analysis. In each case, failure is more easily observed after the operations than beforehand.

Wrong quantitative representation: Most quantitative methods, including the SOM, process numerical vectors. If the data is given in symbolic and structured form, it must be represented in vector form, coded by numerical values. Choice of vector coding is an innovative process, which itself may lead to trial and error.

Semantically unclear data: Data might not be semantically correct for the analysis. Even if data is in vector format, the vectors may not represent the cases to be analyzed. For example, if different vector components represent different semantical categories, the meaning of the whole vector is often unclear, and any grouping of vectors is certainly hard to understand.

Incorrect variables for analyzing method: The purpose of multivariate methods is to find relationships between variables when each variable has an effect to results. Problems occur if the statistical behavior of the variables does not match with the method. Then the analysis will not show the behavior of the data, but some aspects of its coding.

2 Self-Organizing Map as a reduction method

To be more concrete, we select the data reduction to be a set of statistical methods. Our main algorithm, which has best software support, is a tree structured variant of SOM, the use of which is depicted in Fig. 2.

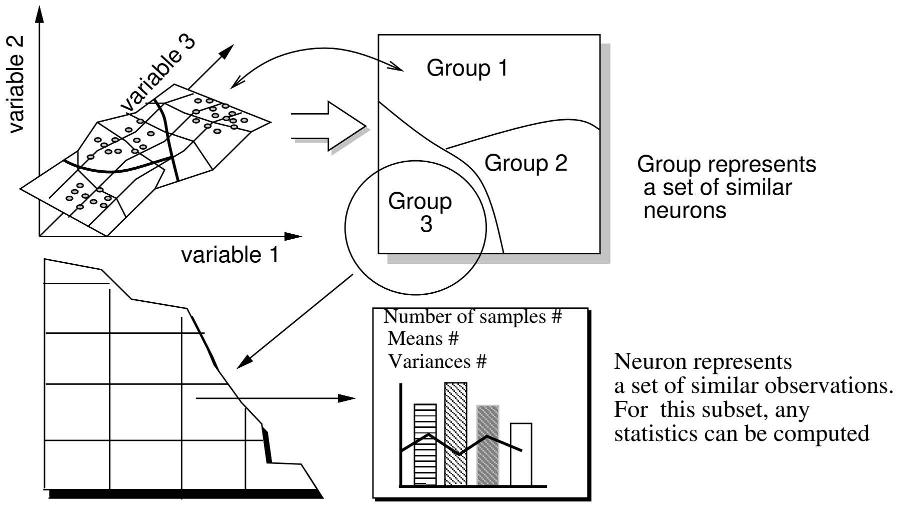

Figure 2: The idea of data representation and visualization with the self-organizing map. A two dimensional surface (SOM) is fitted to go through the middle of a data set. Surface is made of nodes (neurons) and the data vectors are divided between them. Features from multivariate data can then be computed and visualized on the two dimensional surface.

The SOM is seen as a projection method which maps high-dimensional data space to a low-dimensional space, typically to one or two dimensions. At the same time it is also a clustering

algorithm, a topology-preserving mapping that maps similar data vectors to near-by neurons, (more information about SOM can be found from references [2] and [3]).
This provides some promising properties for data analysis. When the number of neurons is large enough, the target clusters are not met directly, but as a result of a set of neurons that themselves represent sets of data vectors. Thus SOM works as a subsymbolic representation where the "sub-meanings" of the prototypes emerge quite easily, even if the clustering objective is fuzzy in the beginning. At the same time the number items is reduced to much smaller set of prototypes, which themselves are ordered according to their similarity. This is an important aid for visualization, and allows the user to see a more complete picture of the data.
The Tree-Structured Self-Organizing Map (TS-SOM) [4] is a variant of SOM having several SOMs in a tree structure with different resolutions. In the context of data analysis TS-SOM provides some additionally useful properties. The tree structure improves the computational effectiveness, allowing very large data sets to be analyzed interactively. It also eliminates the need of parameter adjustments, such as the size of the neighborhood, giving a direct meaning for the weighting of neighbor connections as a smoothing process. Very satisfying results can be achieved with fixed default parameters, making the methods suitable for applicants without deeper expertise in neural computing. The multiple resolutions of SOMs eliminates difficulties in the selection of size of network, and gives the representative power of hierarchal clustering methods.

3 Visualization

Although the self-organizing map is only one of our data reduction methods, it is the most important, it has a specific support for visualization, and thus it deserves special attention. Only some of the visualization techniques can be described here. These are demonstrated via visual presentations of the well-known Boston data, that contains data records from sold apartments in Boston area, which cannot be described further in this context.
Now, every neuron captures a set of Boston apartments. In addition to usual visualization, any statistical values computed and related to the neurons can be visualized. Usually such a statistics is computed directly from original data, making the values easy to interpret. The neurons can be presented as graphical objects in 3-dimensional space, where the places of neurons are set according to the topology or any computed variables, see for example Fig. 3.

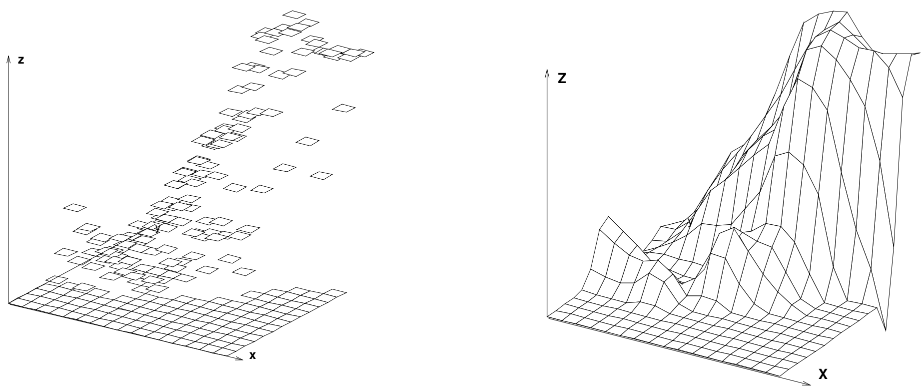

Figure 3: Two SOM presentations in 3-dimensions. Neuron places in xy-plane are related to the SOM topology and the value of field "apartment size" is mapped on the z-axis. Left figure shows the neurons, while the topology surface is shown on the right.

The interpretation of results can be facilitated by creating visual features such that as many dimensions as possible can be visualized at the same time. Presentation may include, for instance,

gray levels, sizes of neuron objects, bars, line diagrams, textual labels inside neurons, and contours, as depicted in Figs. 4 and 5.
Any graphical presentation may use any data. For example, textual labels may be triggered as a function of variable values, or they can be best matching record names to the neurons. Neuron sizes can be selected according any values, for example to illustrate the distances between neurons. In practice, only the readability of presentation restricts the number of simultaneous graphical features. Yet, it is also possible to create several views to the SOM and link them together.

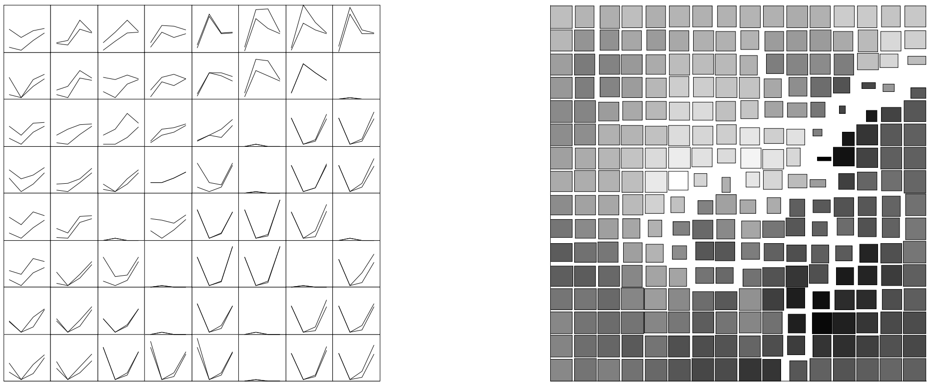

Figure 4: Left figure: The line diagrams display minimum and maximum values of four variables in every neuron: "degree of industry", "plot size", "distance from the city" and "air pollution". Right figure: The detailed U-matrix presentation [1] computed over all the weights in neurons. The distances between the boxes of neurons describe the dissimilarity of neurons. In addition, the gray level of neurons describes the number of rooms.

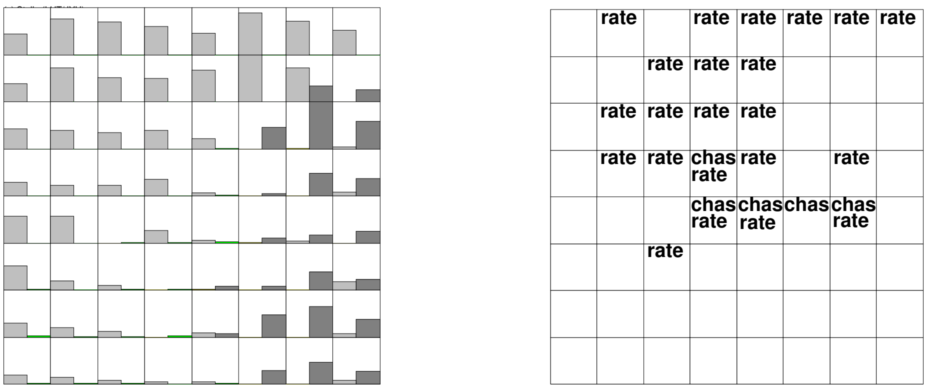

Figure 5: Left figure: Two variables of neurons presented by bars: the first bar describes the number of crimes in residential area and the second bar the distance from the city. Right figure: The field labeling displays the label of field in neuron if a corresponding value fulfills a given rule. Here the rule for label "rate" is the average price in range 25-100 k$ and for field "chas" is that over 50% of apartments are on river side.

4 The Neural Data Analysis environment

The use of SOM and its visualization seems quite straightforward, unless it is examined in close detail under the general framework of data analysis. Here a typical analysis sequence could be as follows:

1. Select data for training.
2. Build the TS-SOM and train it.
3. Classify data into groups of records, which are best matching records for the neurons. The data set does not have to be the same one as in the training but its fields must have a corresponding meaning.
4. Select data for visualization. This could be any data that has corresponding records with the classified data.
5. Compute features that represent the groups of selected data for visualization.
6. Visualize the computed information with help of the TS-SOM structure.

It is easy to see that this type of sequence can be very complicated to realize unless there is a supporting environment for it. In our earlier experiments similar sequence, when made with separate computer programs under a standard file system could take a whole week of manpower. Especially data selections, preprocessing, and transformations between different data formats and software tools were very costly.
As a result the interactive data analysis, where the user can test, verify, reject, and repeat operations on quick manner became impossible. As a solution the NDA environment was designed as a standardized environment to bring the time of the analysis from days to minutes. In some parts it is similar to other analysis programs like Matlab, S+, and SAS, but it has special support for the structures and visualization of the TS-SOM algorithm.

4.1 The basic data types

The aim was to find a compact set of basic operands that support generic data analysis, while keeping the operations user friendly. This requires a compromise between the generality and information hiding from the user: when information and operations are hidden from the user the structures and operands are less generic.
The set operands was selected after careful consideration, it is a quintuple Γ:$< \mathbf{f},\mathbf{d},\mathbf{c},\mathbf{g},\mathbf{s} >$, whose members are:

f *(field)* Field is a vector of data that can include values with any basic data types, integers, floating points or strings. All the values must have the same type in a field, and a single value can be accessed by index.

d *(data frame)* Data frame is an ordered set of fields. The user may build frames by simply naming the fields to be included.

c *(classified data)* Classified data is a set of variable length vectors of indexes. It is used for selecting the records from data fields into groups. For example, if i is an index of a neuron then $\mathbf{c}_i$ could be a vector of record indexes that are classified to that neuron.

g *(graphic)* Graphic is a specific structure that includes graphical specification and representation of selected structures and data.

s *(structure)* Structure is any specific collection of data. For example, TS-SOM has a specific data structure that includes structured information of the neural network, values of the prototypes (weights), and operational parameters.

4.2 Operations

Operations in the NDA environment are simply transformations between the data types. Some specific data structures like graphics can be updated by successive operations, while the data frames and fields are typically not modified after creation.

It is not possible to give a complete set of operations here. Instead a brief list of example different types operations is given. In the list the user given parameters are denoted as an additional operand $\mathbf{p}$.

`select:` $\mathbf{f}_1 \times \mathbf{f}_2 \ldots \mathbf{f}_n \rightarrow \mathbf{d}$. Operation builds a data frame from a given set of fields.

`equalize:` $\mathbf{d}^{old} \times \mathbf{p} \rightarrow \mathbf{d}^{new}$. Equalization takes a data frame and equalizes the variation of variables according to given rule.

`somtr:` $\mathbf{d} \times \mathbf{p} \rightarrow < \mathbf{s}^{som}, \mathbf{d}^{weights} >$. SOM training creates the TS-SOM structure and organizes it. The operand $\mathbf{p}$ involves optional parameters for defining the structure and the training algorithm of TS-SOM. Weights are stored separately in an ordinary data frame, $\mathbf{d}^{weights}$.

`somcl:` $< \mathbf{s}^{som}, \mathbf{d}^{weights} > \times \mathbf{d} \rightarrow \mathbf{c}$. Operation classifies data records with the TS-SOM. The result is a classified data $\mathbf{c}$.

`clavg:` $\mathbf{d} \times \mathbf{c} \rightarrow \mathbf{d}^{stat}$. Operation computes the expectation of all data subsets, specified by the classified data frame $\mathbf{c}$, and stores the result as a new data set. This can be used, for example, to compute neuron specific features for visualization.

`mkgrp:` $\mathbf{s}^{som} \times \mathbf{d}^{stat} \rightarrow \mathbf{g}$. Graphic is handled via a special data structure $\mathbf{g}$ that is also stored in the name space. This operation just creates one graphic structure. There are numbers of operations for computing graphical features from data.

5 Conclusions

The main advantage of the NDA environment is the support for a specific neural network structure. Information is easily transferred between the neural algorithms and other data sources, making a large variety of data analysis operations easy to realize.
Additional benefits follow from its implementation which has two parts: a kernel and an application client. The NDA kernel includes all the functionality of the system, while the client is an application specific interface to it. All data handling operations, including visualization can be controlled via a command interpreter, which can actuated from the application client. It should also be noted, that the implementation of the kernel is machine independent ANSI-C code, while the application client needs a small set of environment specific routines for graphics and user input. Customized clients that provide user friendly interfaces for predefined analysis sequences are effortless to build.

References

[1] Jukka Iivarinen, Teuvo Kohonen, Jari Kangas, and Sami Kaski. Visualizing the clusters on the self-organizing map. In Christer Carlsson, Timo Järvi, and Tapio Reponen, editors, *Proc. Conf. on Artificial Intelligence Res. in Finland*, number 12 in Conf. Proc. of Finnish Artificial Intelligence Society, pages 122–126, Helsinki, Finland, 1994. Finnish Artificial Intelligence Society.

[2] Teuvo Kohonen. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[3] Teuvo Kohonen. *Self-Organizing Maps.* Springer, Berlin, Heidelberg, 1995.

[4] Pasi Koikkalainen. Progress with the tree-structured self-organizing map. In A. G. Cohn, editor, *Proc. ECAI'94, 11th European Conf. on Artificial Intelligence*, pages 211–215, New York, 1994. John Wiley & Sons.

[5] Matthew B. Miles and A. Michael Huberman. *Qualitative Data Analysis.* SAGE Publications, CA, USA, 1994.