

KKAnalysis - A software for unsupervised classification by Self Organizing Maps and Cluster Analysis

Extended Documentation

A. Messina¹, H. Langer²



¹*Istituto Nazionale di Geofisica e Vulcanologia
Sezione Roma 2
Via di Vigna Murata 605, 00143 Rome, Italy*

²*Istituto Nazionale di Geofisica e Vulcanologia
Sezione di Catania
Piazza Roma 2, 95123 Catania, Italy*

SUMMARY

1	INTRODUCTION.....	3
1.1	OBJECTS, CLASSES, FEATURES, PATTERNS.....	3
1.2	METRICS	3
1.3	CLUSTER ANALYSIS S. STR.	4
1.4	THE SELF ORGANIZING MAP.....	6
2	KKANALYSIS.....	7
2.1	PRELIMINARIES	7
2.2	INSTALLATION	8
2.3	FILES.....	9
2.3.1	<i>INPUT FILES</i>	9
2.3.2	<i>OUTPUT FILES</i>	9
2.4	GETTING STARTED	10
2.4.1	<i>THE “INPUT FILE” FRAME</i>	10
2.4.2	<i>THE “FIGURES” FRAME</i>	11
3	CONFIGURING KKANALYSIS – THE “SETTINGS”	23
	APPENDIX.....	27
	REFERENCES.....	29

1 Introduction

1.1 Objects, classes, features, patterns

Classification can be understood as a mechanism to assign objects to a category or class. Objects are characterized by a number of features which can be metrical, ordinal or nominal data. A set of features is represented in a feature vector. An object characterized by a feature vector is called a pattern. The choice of the features depends essentially on practical considerations and is governed by two rules :

- (i) The features should allow to identify the objects uniquely.
- (ii) The shorter the feature vector the better it is.

Sometimes features can be gained directly from the description of the object, but frequently preprocessing, such as normalizing or transformation of the information is necessary in order to meet these goals properly. Here we consider feature vectors made up by metric data, which is the easiest situation to handle but fortunately also among the most frequent ones. In particular, metrical data allow to define dissimilarities between objects which is important for unsupervised classification, to which our software addresses.

Unsupervised classification is often referred to as clustering. In clustering classes are formed by culling together patterns which have a minimum distance among each other. Contrary to supervised classification, where the classification problem is learned from examples of feature vectors with known class membership, the a-priori information used here resides in the definition of a metrics representing the distance between patterns.

1.2 Metrics

All metrics must have four properties. Given the feature vectors **a**, **b**, **c** and $d(,)$ being the distance between two vectors these properties are

- Nonnegativity : $d(\mathbf{a}, \mathbf{b}) \geq 0$
- Reflexivity : $d(\mathbf{a}, \mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$
- Symmetry : $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$
- Triangle Inequality: $d(\mathbf{a}, \mathbf{b}) + d(\mathbf{b}, \mathbf{c}) \geq d(\mathbf{a}, \mathbf{c})$.

In the literature various options are reported, such as the Euclidean distance, the Mahalanobis distance, Manhattan (or city block distance) etc. More about these definitions can be found in textbooks like Duda et al. (2000). In our toolbox KAnalysis we use the Euclidean distance, in particular in the context of the so-called “Self Organizing Maps” (SOM). Euclidean distances are also used in “K-Means” cluster analysis and “Fuzzy C-Means” clustering. It is defined by

$$d = \|\mathbf{a} - \mathbf{b}\|^{1/2}$$

The homogeneity of a cluster of patterns is expressed as the sum of the squared distances of all patterns with respect to the cluster centroid vector, i., e.,

$$D = \sum_i d(\mathbf{a}_i, \mathbf{m})^2$$

where **m** is the centroid vector of the cluster, **a_i** is the *i*-th feature vector, and *i* runs from 1 to N_k , N_k being the number of patterns belonging to the *k*-th cluster. In unsupervised classifying we seek to

form most homogeneous clusters, which is minimizing D . The use of a metrics based on the Euclidean distance implies that all components of $\mathbf{a} - \mathbf{m}$ have the same weight, consequently the hull of the clusters tend towards a spherical (or “hyperspherical”) shape.

Assuming that the patterns scatter around a centroid \mathbf{m} following a certain statistical distribution – let’s say (multivariate) Gaussian - the distance $\mathbf{a} - \mathbf{m}$ becomes a measure of probability that a pattern vector \mathbf{a}_i belongs to a cluster k with its centroid \mathbf{m}_k . We prefer to assign \mathbf{a}_i to cluster k instead of cluster l if

$$\mathbf{a}_i - \mathbf{m}_k < \mathbf{a}_i - \mathbf{m}_l$$

The Euclidean distance works fine as long as there is no correlation between the components of the pattern vector, but becomes misleading if significant correlation exists (see Fig. 1). In the latter case the patterns form elongated clouds around the cluster center. Using concepts of the Principal Component Analysis we may define a more general distance, i. e.,

$$d = \|(\mathbf{a} - \mathbf{m}_k) \mathbf{C}_k^{-1} (\mathbf{a} - \mathbf{m}_k)^T\|^{1/2}$$

where \mathbf{C}_k is the covariance matrix of all patterns belonging to the k -th cluster. Applying \mathbf{C}_k^{-1} we transform the original feature vector into a new feature space by rotating the patterns in a way that correlation among the components vanishes, and normalize with respect to the marginal variances in the new system of axes. We then may assign class memberships using Euclidean distances calculated on the base of the transformed feature vectors.

1.3 Cluster analysis s. str.

Cluster analysis can be distinguished in hierarchical and non-hierarchical partitioning techniques. From hierarchical cluster analysis one obtains dendrograms, in which subgroups are formed dividing “parent groups”, i.e., clusters generated in an earlier step of the analysis. The hierarchical strategy is justified in fields like bio-taxonomy, where the a-priori assumption of a hierarchical data structure has sound scientific reasons. Often this is not the case, however, so we prefer the non-hierarchical, partitioning strategy of clustering. In this case the number of desired clusters has to be chosen a-priori. Every time the number of desired clusters is changed, the whole clustering process restarts. Consequently, the results obtained for different numbers of clusters are independent of each other.

As stated earlier the task of cluster analysis is to find a partition into groups having optimum homogeneity. In “K-Means clustering” the global measure of homogeneity of the clusters is given by the sum of the squared Euclidean distances found in all clusters, i.e.,

$$D = \sum_k \sum_i d(\mathbf{x}_i, \mathbf{m}_k)^2 = \sum_k \sum_i (\mathbf{x}_i - \mathbf{m}_k) (\mathbf{x}_i - \mathbf{m}_k)^T$$

where k , for M clusters, runs from 1 to M , and i runs from 1 to N_k , N_k being the number of patterns \mathbf{x} belonging to the k -th cluster. An optimum partition is achieved when D is minimum. Though K-Means clustering is a widely used method, it may suffer from the drawback that all components of the feature vector are considered as linearly independent, i.e., are not correlated to each other. In multivariate statistics this problem is fixed by using the Mahalanobis distance mentioned above, using the inverse covariance matrix in order to adjust the metric:

$$d = \|(\mathbf{x} - \mathbf{m}_k) \mathbf{C}_k^{-1} (\mathbf{x} - \mathbf{m}_k)^T\|^{1/2}$$

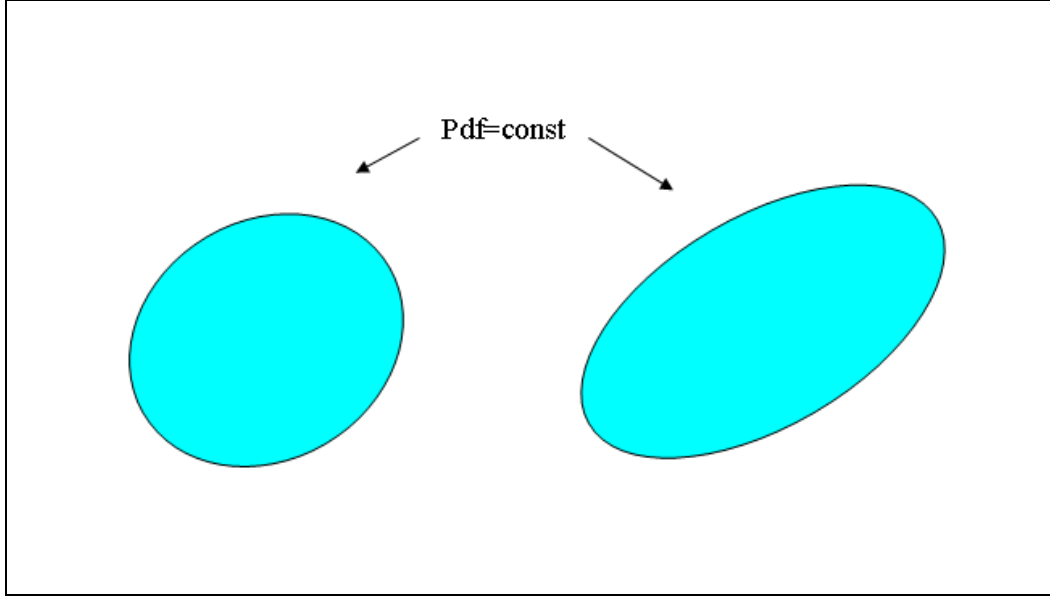


Fig. 1. Cluster hulls and probability. Hulls are spherical for uncorrelated features and elliptical for features with correlation. The hulls can be understood as isolines of probability density, turquoise areas represent confidence spaces for the probability that a cluster member can be found within the hull

However, we cannot use directly the sum of squared Mahalanobis distances for identifying an optimum partition as we are loosing, due to the term \mathbf{C}_k^{-1} , information about the absolute dispersion of the cluster. A way out of this dilemma was proposed by Späth, 1983, where \mathbf{C}_k^{-1} is replaced by $\mathbf{G}_k = \det(\mathbf{S}_k)^{(1/r)} \mathbf{S}_k^{-1}$. In this expression \mathbf{S}_k is the dispersion matrix of the k -th cluster, and r is the rank of the dispersion matrix. Consequently, the optimum partition is now found by minimizing

$$D = \sum_k \sum_i (\mathbf{x}_i - \mathbf{m}_k) \mathbf{G}_k (\mathbf{x}_i - \mathbf{m}_k)^T$$

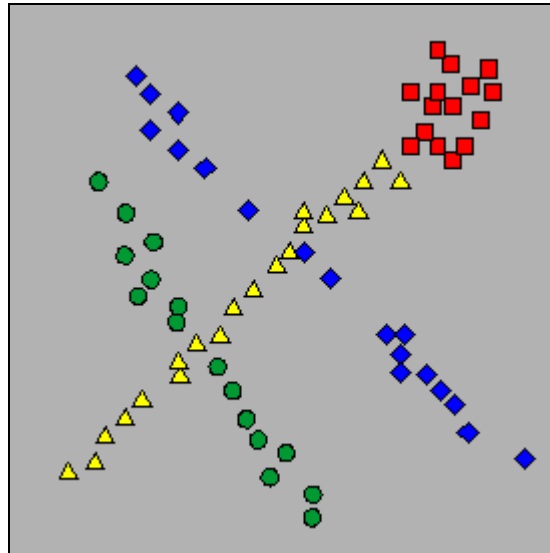


Fig. 2. Clusters obtained with the adaptive determinant criterion

This measure of homogeneity of clusters is known as adaptive determinant criterion, and accounts for groups of data with different directions of their principal axes (Fig. 2). Note that the clusters in the demonstration data set of Fig. 2 may even penetrate each other. Nevertheless, the partition is not

fuzzy, as each object belongs exclusively to one cluster. We address the interested reader to textbooks of cluster analysis like Anderberg (1973), Späth (1983, 1985) and Duda et al (2000).

1.4 The Self Organizing Map

The Self Organizing Maps (henceforth SOM) were invented by Teuvo Kohonen (Kohonen, 1984) and are often referred to as Kohonen maps. They provide a way of representing multidimensional data in much lower dimensional spaces than the original data set. The process of reducing the dimensionality of vectors corresponds to a data compression technique known as vector quantization. Kohonen's method creates a network that stores information in such a way that it maintains the topological relationships within the patterns of the data set.

SOM are made up of a number of nodes to each of which a weight vector \mathbf{W}_i is assigned (Fig. 3a). The \mathbf{W}_i have the same dimensionality as the original feature vectors. SOM are constructed in an iterative procedure, considering the differences

$$D_{ij} = \sqrt{(\mathbf{W}_i - \mathbf{V}_j)^T (\mathbf{W}_i - \mathbf{V}_j)}$$

between the normalized input vector \mathbf{V}_j and the weights \mathbf{W}_i of the nodes. A core step is the identification of the closest node to the actual input vector, i.e., the best matching unit (BMU) for the j -th pattern. Throughout this identification process, neighboring nodes lying within a certain radius of influence are considered as well. Once BMU and the nodes falling within the area of influence are identified, the weights are gradually adjusted according to the so-called learning rate λ . The rate to which the weight of the nodes are adjusted decreases with the distance Δ between each node and the BMU. The upgrade of weights follows the following relationship

$$\mathbf{W}_i(t+1) = \mathbf{W}_i(t) + \varphi(\Delta, t) \cdot \lambda(t) \cdot D_{ij}(t)$$

where φ describes the distance dependence of the upgrade of a node. $\varphi(\Delta, t)$ is maximum for the BMU, whereas nodes outside the radius of influence are not upgraded at all (Fig. 3b). During a cycle of training this procedure is repeated for all input vectors. To achieve the stabilization of the map, both learning rate and radius of influence decrease during each iterative cycle. Eventually, the map will depict the input vectors, the BMU, and the weight vectors calculated.

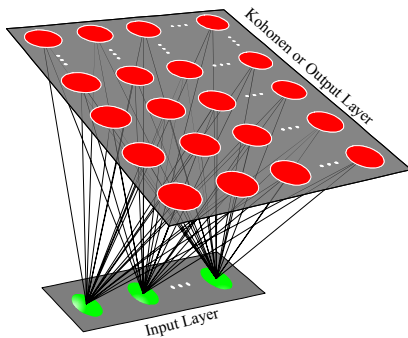


Fig. 3a. A bidimensional SOM

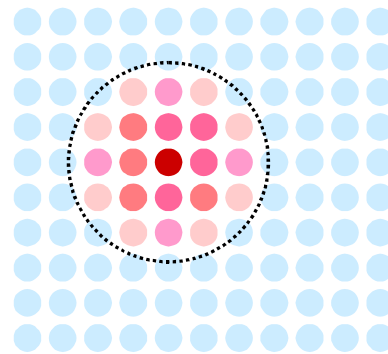


Fig. 3b. BMUs and area of influence

SOM become particularly instructive when the weights are represented with a color code (Fig. 3c). For this purpose we first apply Principal Component Analysis, transforming the weights in a two-dimensional (2D) representation space made up of the principal axes z_1 and z_2 . z_1 and z_2 are normalized to a range of $[0 \dots 1]$. A third value z_3 is obtained either as $1 - z_1$ or $1 - z_2$. Then, we assigned the principal colors: red, green, and blue to each of the z_i , e.g., red to z_1 , green to z_2 , blue to

z₃. Each node receives a mixture of the colors according to its original weight, respectively to the projection of the weights in representation space. The famous World Poverty Map (see <http://www.cis.hut.fi/research/som-research/worldmap>) gives an immediate impression of the power of this representation technique.

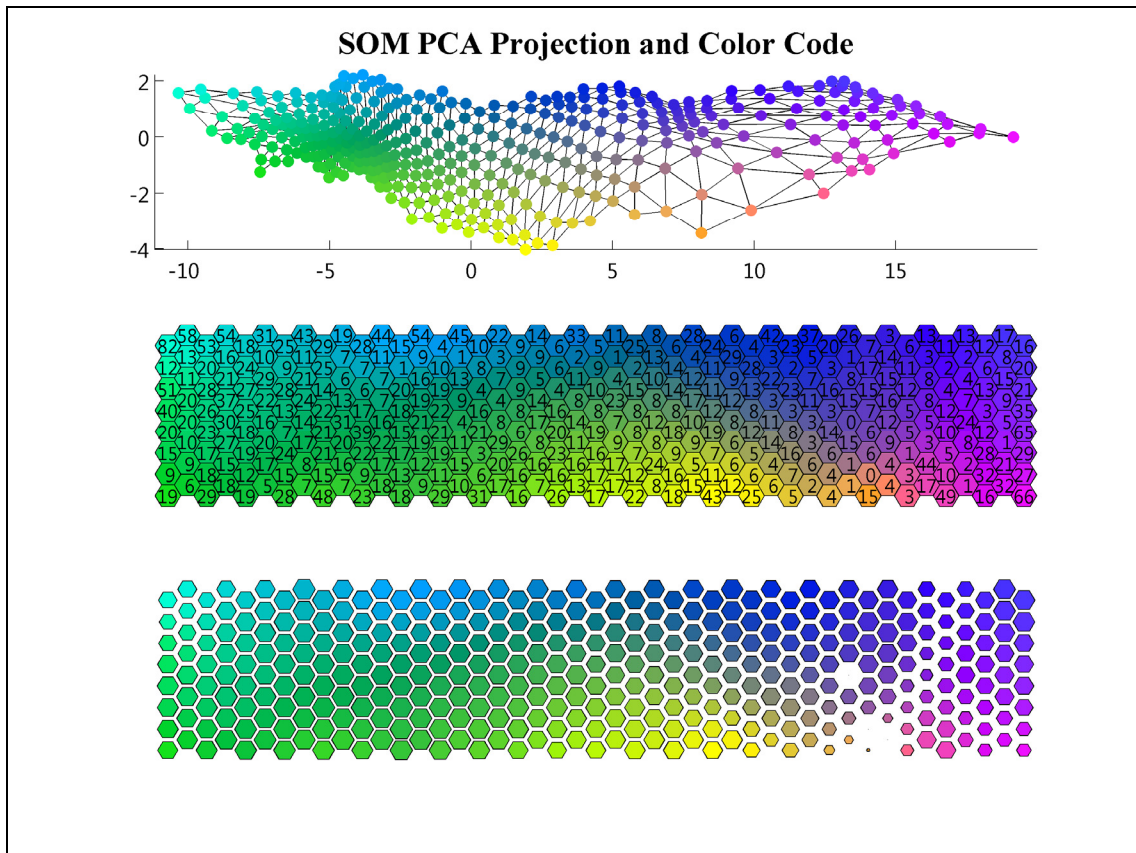


Fig. 3c. Color Coding of the SOM. The plot above gives the position of the nodes in the 2D representation space. The graph in the middle shows the number of patterns represented by the nodes of the SOM. The nodes can be understood as small clusters. Note that nodes close to each other in the map have similar colors and are also close to each other in the original feature space. We can say that a well defined SOM shows a “topological fidelity”. The graph at the bottom represents the homogeneity of clusters given by the nodes: the larger their size the lower their internal dispersion

2 KKAnalysis

2.1 Preliminaries

KKAnalysis is a collection of methods for unsupervised classification and clustering. The software package exploits widely routines of the SOM Toolbox 2 for MATLAB (Vesanto et al., 2000, see <http://www.cis.hut.fi/projects/somtoolbox/>). Besides functions related to the SOM, we have been adding K-Means clustering, available in the toolbox, the MATLAB fuzzy clustering as well as adaptive determinant clustering, the latter taken from Späth, (1983). All clustering schemes can be applied to the map created by the SOM routines and to the original patterns as well. KKAnalysis provides both numerical and graphical output.

In order to render KKAAnalysis user-friendly a specific GUI has been created where every parameter is easily configurable by a click on the appropriate control (buttons, check boxes, text boxes, etc.). Moreover, each of them has a so called tooltip text which describes in few words the function of the control.

KKAnalysis was tested on x86 architecture with Microsoft Windows XP™, Microsoft Vista Business™ and GNU/Linux Ubuntu 9.04. A reasonable performance was achieved on a machine

with a Pentium 4 processor with a 3 GHz clock frequency, and 1 GB RAM. The software itself requires a storage capacity on the hard disk of ca. 1.5 MB. Note, however, that KKAnalysis allows to save the graphical output in high quality, ready to use in publications. So a large mass of data can accumulate rapidly.

2.2 Installation

KKAnalysis was developed under MATLABTM. This programming language requires a “virtual machine” (to say it in terms common to Java users) called MCR (MATLAB Component/Compiler Runtime) for the execution of deployed applications. If MATLAB is not available, the MCR (v.7.7 or higher for Windows or v.7.9 version for Linux) must be installed in order to run KKAnalysis. Both MCR versions, together with KKAnalysis executables and auxiliary material, can be downloaded from the digital archive <http://www.earthref.org/>. Note that MCR installer for Linux has been splitted into two parts in order to upload it in the archive (the limit for a single file is 200 MB). The MATLAB Component Runtime installation on Windows system can be carried out clicking “MCRInstaller.exe” file and following the wizard. Under Linux systems open a terminal window and make sure that the “Bash” shell is running. Bash is the default on Ubuntu. Enter the directory to which both of the “MCRInstaller.bin” parts were downloaded. Then type the commands “cat MCRInstaller-part* > MCRInstaller.bin” to unify the parts and “chmod 777 MCRInstaller.bin” to make the file executable. Finally, type the command “./MCRInstaller.bin -console” and follow the command line guided procedure. Note that you may need the superuser privileges to perform this installation in the default directory (“/opt/MATLAB/MATLAB_Compiler_Runtime” on Ubuntu). If this is not desired choose an alternative destination for the installation, where read-write-execute permissions are available.

As MCR is included in MATLAB environment, no extra installation is necessary when a MATLAB is present on your computer. The MATLAB version should be R2007b or higher for Windows operating systems, and R2008b or higher for Linux.

On Windows systems, the installation of the program can be carried out clicking “KKAnalysis_setup.exe” file. The only thing the user has to do is to accept the license conditions. KKAnalysis can be invoked either from the “Programs → KKAnalysis” item or by clicking the shortcut on your desktop (if you allowed for this option during the installation process).

In Linux environment KKAnalysis does not require any installation. Simply extract “KKAnalysis.tar.gz” file to a directory of your choice. The only thing you have to do is setting the correct MCRROOT variable inside the “KKAnalysis.sh” bash script. On computers with MATLAB (see the requirements above), you should set MCRROOT to the root path of the MATLAB installation directory (for example, MCRROOT=/bin/MATLAB).

If you have installed the MCR downloaded from the archive, MCRROOT should be set to installation path adding “/v79” (the extension “v79” reports the MCR version, 7.9 in our case). For instance, if the MCR was installed in the default directory (i.e., “/opt/MATLAB/MATLAB_Compiler_Runtime”, see above), edit “KKAnalysis.sh” and set “MCRROOT=/opt/MATLAB/MATLAB_Compiler_Runtime/v79”. Finally you can launch KKAnalysis typing “./KKAnalysis.sh”. Note that it could be necessary to disable Compiz (or Beryl) graphical effects on Ubuntu in order to guarantee KKAnalysis properly execution.

A final request : READ THE LICENSE CONDITIONS CAREFULLY AND RESPECT THEM.

2.3 Files

2.3.1 Input Files

The input files of KKAnalysis concern the data used in classification.

Header	Kopf	Testa	Hair	Hat	
Header	Kopf	Testa	Hair	Hat	
red	4.927	4.953	4.568		1
green	4.872	4.905	4.52		3
blue	4.85	4.916	4.594		5
yellow	4.821	4.861	4.516		6
grey	4.959	4.975	4.604		7
black	4.996	5.03	4.686		9

Tab. 1 A typical KKAnalysis input file, with headers row, label columns, and a specific column for the abscissa of the graphical output

The data file must be provided by the users. In its typical form it consists of rows and columns, where a row contains a feature vector of a pattern. See the example above (Tab. 1). The area highlighted in turquoise delineates such a data file. Sometimes a data file (see the example shown above) may contain additional information which, though being useful for the user, are not exploited for classification purposes. For instance, there may be descriptive rows at the top, the so called “header”. There may also be labels for each pattern, written in the first columns of the data file (in our example the labels “red”, “green”, “blue”, etc.). KKAnalysis automatically recognizes these non-numeric rows and columns and doesn’t consider them in the analysis. If desired, one of the columns (purple, in our example) may be exploited for the creation of the abscissa in the graphical representation of the results. If this is desired make sure that the corresponding column contains only integer values.

2.3.2 Output Files

Files created by KKAnalysis can be distinguished in two types : log files and files reporting the results of a session. In the log KKAnalysis reports controlling parameters used during a session, producing a sort of “execution history” related to the various runs carried out during a session. The log file is created by the program at the beginning of each session, i.e., every time when KKAnalysis is started. Its name is “KKA_Log_YYYYMMDDThhmmss”, where YYYYMMDD is the date (year/month/day) and hhmmss is the time (hours/minutes/seconds), both of them referred to the beginning of the session.

The settings used during a specific run can be saved in a configuration file, the name of which is specified by the user. A configuration file created during an earlier session can be reloaded in order to reproduce the corresponding classification results. However, KKAnalysis automatically reloads the setting values of the previous session at the beginning of a new one. More details on the configuration parameters are given below.

Besides this, KKAnalysis creates files storing the results of each session in alphanumeric form using ASCII standard. Files “KKA_Results_YYYYMMDDThhmmss” (indicating date and time of execution) generally, contain 5 columns, reporting (i) the index of the pattern, (ii) the cluster membership, (iii,iv,v) the RGB color code of the BMU to which the patterns belong. Another file, named “KKA_Node_Weights_YYYYMMDDThhmmss”, has as a first column with the indexes of the map node, than a sequence of columns with their corresponding weights. The number of weight columns corresponds to the number of components used in the classification. Last but not least, KKAnalysis permits saving the graphical output in the format “TIF” as well as in MATLAB

format. The latter is assigned to an extension “*.fig”. More about these files in chapter 2.4.2 where the graphical output is described in more detail.

2.4 Getting started

The following descriptions and all the images of this documentation are referred to the Windows version of KKAnalysis. However, the Linux version essentially presents the same features. Click on the desktop shortcut “KKAnalysis” (or use the corresponding item in your Program Files menu). KKAnalysis prompts you the subsequent “Welcome” sheet (Fig. 4a) offering various options. If you are running the program for the first time, all fields in the sheet will be empty. Otherwise, all fields report the choices of the last session, which you can re-use if you want. In the “welcome” sheet there’s also a “Console” which is used by KKAnalysis to prompt some information during run time, permitting the user to check whether the program is working properly.

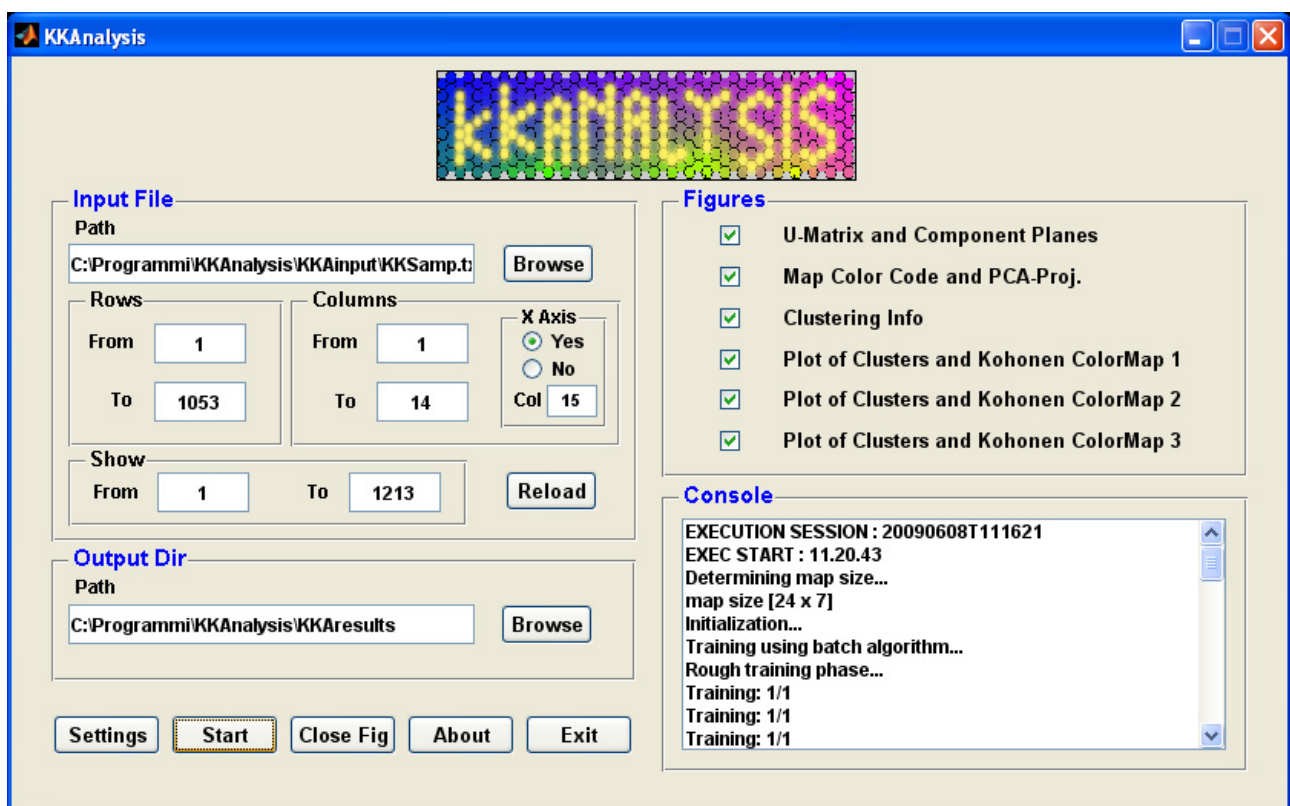


Fig. 4a. Welcome to KKAnalysis!

2.4.1 The “Input File” Frame

The “Input File” frame (Fig. 4b) contains four subframes: “Path”, “Rows”, “Columns” and “Show”. In the field “Path” you insert the full name (path included) of the raw input data, which should have the format of a matrix, where data are written in rows and columns. The operation of file selection can be carried out either by using the “Browse” button next to the edit box or writing by hand its name.

Fig. 4b. The “Input File” frame

Once identified the right file, KKAnalysis reads it and reports the structure found, i.e., the number of rows and columns encountered. This information is shown in the frames “Rows” and “Columns”. In KKAnalysis the feature vectors of the patterns correspond to the rows in the input file. The number of rows is equivalent to the number of patterns, whereas the number of columns reads as the number of components of the feature vector. As mentioned in “Input Files” paragraph, non-numeric lines and columns inside the file are automatically discarded and not used for classification purposes. The values shown in the fields “From” and “To” (placed inside the frames “Rows” and “Columns”) are referred to the effective range of data matrix inside the whole file. For example, it can be possible to see a “From” value equal to “4” because the first three lines of the file are header rows. The entries in these fields can be modified in order to choose how many of the data columns and rows to be used for the construction of the SOM and for clustering. Note that any part within the data table can be selected, with the only condition that lines and columns neighbor each other. The “Show” and “X Axis” frame contain useful settings concerning the graphical output. The former allows the user to set a range of patterns to bring into focus. In other words, the analysis is done, for example, on the entire dataset whereas the output files and figures contain only the range of patterns chosen by the user. Finally, “X Axis” option can be enabled whenever you want to represent the cluster-ID or the color of the BMU not only one-by-one but as a function of some additional variable, for an index representing time (e., hours, julian day number, etc.) . Note that a column used as “X Axis” must contain integers! Activating this option KKAnalysis will display the clustering results as a function of the values given in the column specified in “Col” field.

2.4.2 The “Figures” frame

Besides storing alphanumerical information, such as weights, class membership IDs etc., KKAnalysis produces a number of graphs helping the user to understand its results (Fig. 4c). On the right hand side of the frame, we may select two graphs related to the SOM, “**U-Matrix and Component Planes**” (Fig. 5) and “**Map Color Code and PCA-Proj**” (Fig. 6).

Fig. 4c. The “Figures” frame

The idea of the U-Matrix is to represent the dissimilarities measured between neighboring units of the SOM. In order to improve the readability of the graph, the units of the SOM themselves make part of U-Matrix as well. At the end the total number of elements in the U-matrix is given by the number of nodes of the map and the number of distances between neighboring nodes. Suppose a grid with a 24×7 SOM units. That means we measure 23 distances in vertical and 6 distances in horizontal dimensions, giving a total of $24 + 23 = 47$ rows and $7 + 6 = 13$ columns. As stated earlier, the nodes of the SOM themselves are represented in the U matrix as well. A distance of an object measured with respect to itself doesn't make sense, so the value reported for the nodes themselves is calculated as the average of distances measured to all neighboring nodes.

Distances in the U-matrix are represented by colors. Dark and blue stand for small, bright to red for large values. An inspection of the U-matrix may reveal an immediate idea about the presence of major clusters among the SOM nodes. These clusters would correspond to extended areas with blue colors prevailing, meanwhile cluster borders are easily recognized by bright areas. An interesting aspect of clusters showing up in the U-matrix is that their hulls may be of more or less arbitrary shape, whereas hulls of conventional cluster analysis necessarily have a rather simple geometry. At the moment, however, we have no practical rule available about how clustering could be established based on the base of the U-matrix.

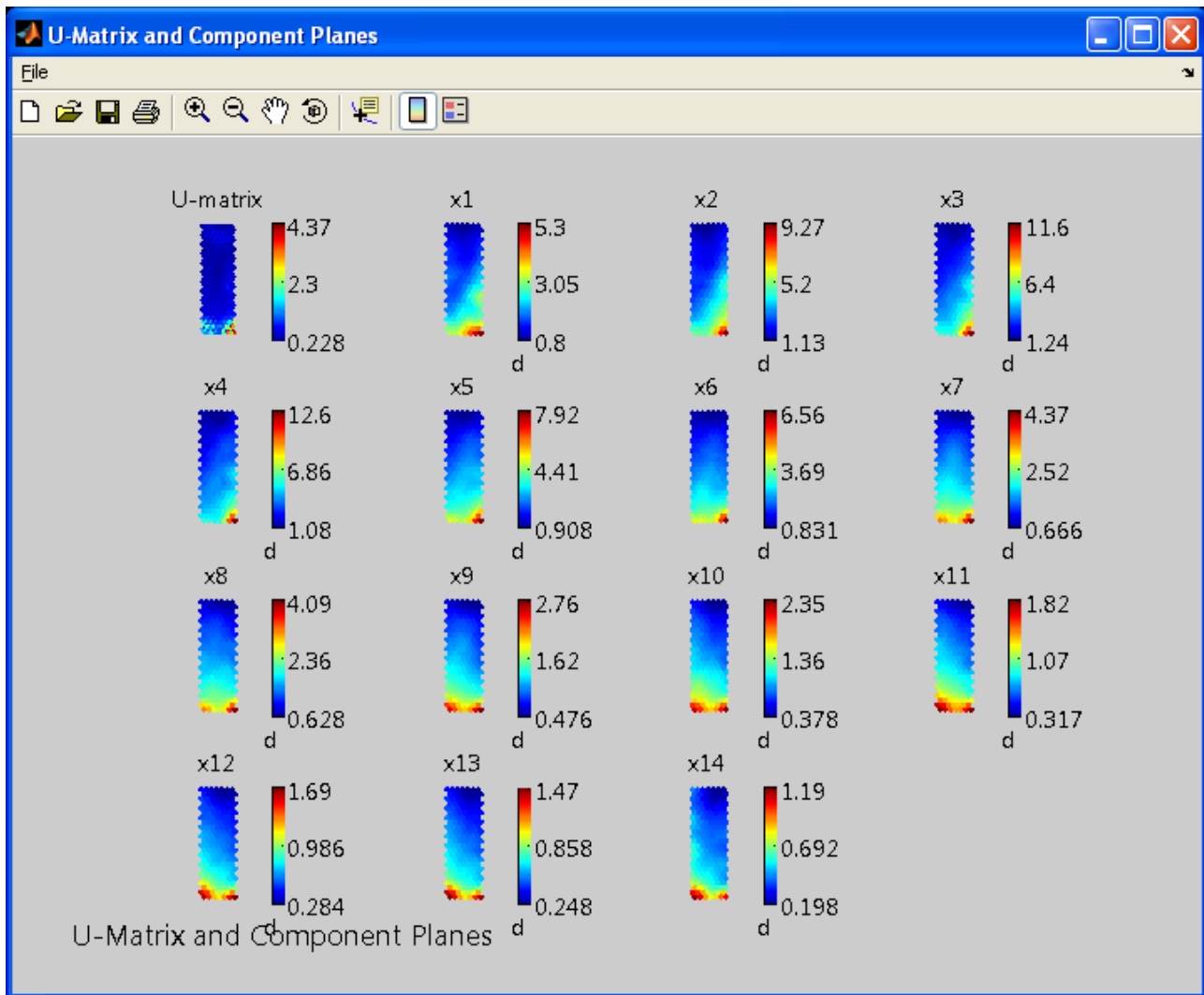


Fig. 5. U-matrix and Component planes. The number of columns of the data file used here is 14

The “Component Planes” represent the position of each BMU with respect to the original variable components. A color code is used. Warm colors (red – brown) stand for large values, dark blue for

small ones. For instance, the BMU at the lower right corner – with a dark brown color - has a position on the x1 axis of the original data space close to 5.3; the exact values can be read in the files “KKA_NodeWeights_” created by KKAAnalysis (the wildcard * stands for date and time of file creation). Note that the Fig. 5 can be saved as MATLAB (*.fig) or TIF file and carries the name “U_matrix_ YYYYMMDDThhmmss, where YYYYMMDD is the date (year/month/day) and hhmmss is the time (hours/minutes/seconds). The “T” in the middle helps to see where the data string ends and day time information begins. Saving of graphical output is activated with the appropriate selections in the “Settings Screen”.

Setting the tick on “**Map Color Code and PCA Projection**”, KKAAnalysis summarizes basic properties of the SOM created during the training process.

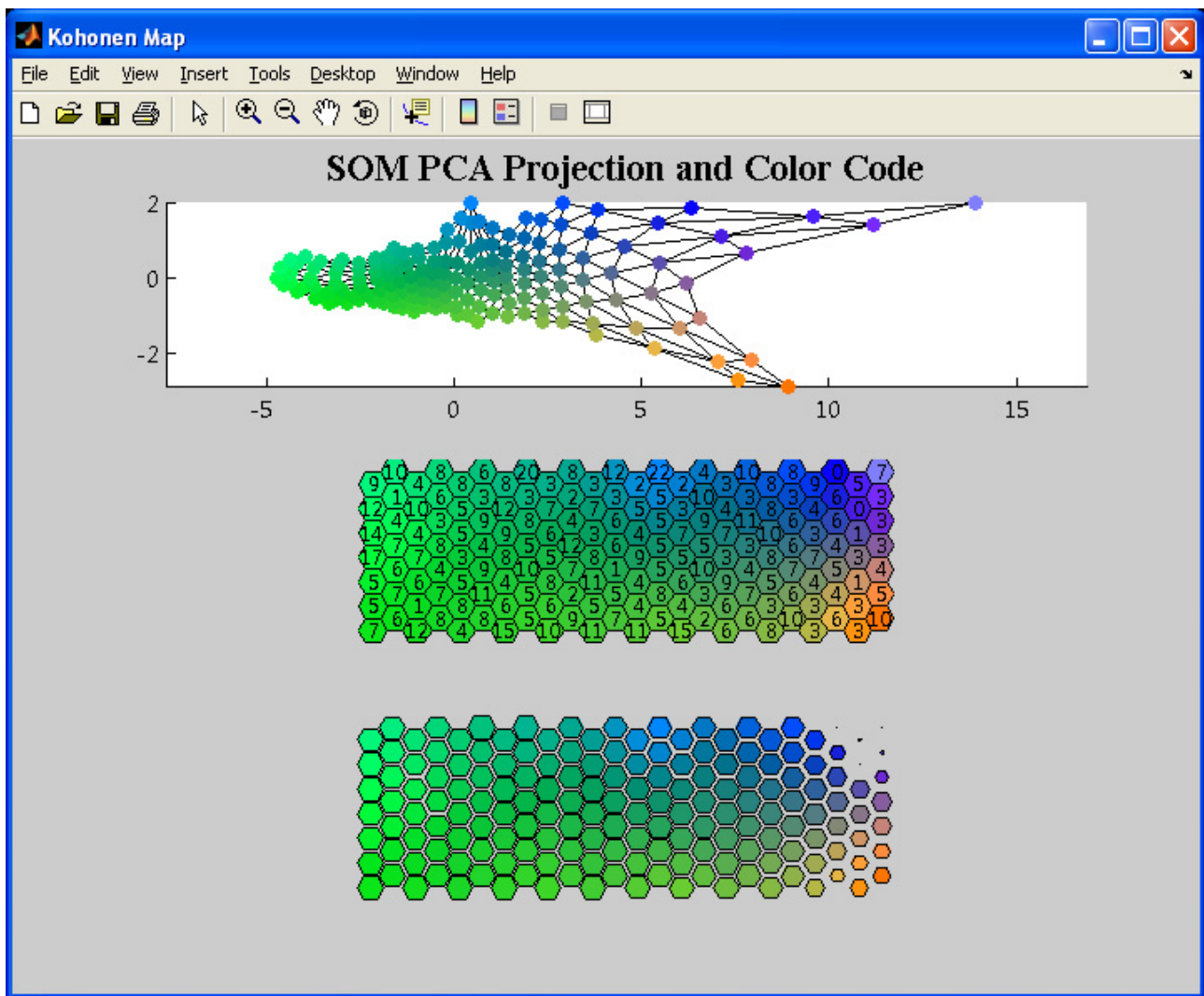


Fig. 6. The SOM using a lattice with a hexagonal topology

Here the program has created a SOM made up of $7 \times 24 = 168$ nodes. For the definition of the map geometry and size KKAAnalysis performs internally a Principal Component Analysis of the covariance matrix of the data vector set. The relation of length/width of the map corresponds approximately to the ratio of the first two eigenvectors.

The uppermost frame in the figure shows the position of each BMU in a system of axes made up by the eigenvectors corresponding to the two largest eigenvalues. Note, however, that eigenvectors and eigenvalues used for the creation of the first frame are obtained from the SOM values instead of the data vectors. We see that the gross of the nodes scatter in a range from ca. 5 to -5 on the abscissa

(that is 10 units in total) and from ca -1.5 to ca 1.5 on the ordinate (i.e., ca. 3 units). By taking aspect the ratio $10/3 = 3.3$ we understand that the geometrical shape spanned by the BMUs, i.e., $24/7 = 3.4$, corresponds nicely to the one made up by the eigenvalues ratio obtained from the covariance matrix of the data.

The second frame shows the color coding of the BMUs. As mentioned earlier the colors of the BMUs are obtained from a 2D PCA of the SOM values, extracting first the two largest eigenvalues plus corresponding eigenvectors of the covariance matrix in order to define a new representation space. In this new space the axes are identified with the principal colors, such as red, and green. The third, blue, for instance, scales with respect to a linear function of one of the two principal components. The position of a node then can be easily inferred from its color, which is a mixture of the three principal colors. Colors of the nodes in all frames are identical. The numbers given for each node corresponds to the number of patterns for which this node was identified as BMU. In our example the green node on the upper left corner of the frame was identified as BMU for 9 patterns. Some nodes turn out to be “loosers” as there were never BMUs, consequently frame 2 reports a “0” for these nodes.

SOM nodes can be understood as microclusters, each attracting and representing a number of patterns. In the third frame of the figure, KAnalysis shows how compact these microclusters are. For the definition of compactness KAnalysis uses the information stored in the U-matrix, i.e., the average of distances separating a node from the neighbouring ones. These large symbols represent compact clusters not sharply distinguished from surrounding ones, small ones indicate clusters with a high degree of heterogeneity. Looser nodes, for which no measure of separation can be defined, are simply marked by a dot.

Note that the Fig. 6 can be saved as MATLAB (*.fig) or TIF file and carries the name “ColCode_YYYYMMDDThhmmss”, where YYYYMMDD is the date (year/month/day) and hhmmss is the time (hours/minutes/seconds). The “T” in the middle helps to see where the data string ends and day time information begins. Saving of graphical output is activated with the appropriate selections in the “Settings Screen”.

The “**Clustering Info**” figure is related to options of KAnalysis for cluster analysis s. str, i.e., K-Means, Fuzzy C-Means, and clustering with the adaptive determinant criterion for the metrics of distance (see Fig. 2). It provides some basic parameters to check the quality of achieved results, helping the user to compare various set ups and to identify the most suitable configuration. As we shall discuss later KAnalysis incorporates three strategies of cluster analysis. The first, default option is the traditional “*K-Means*”.

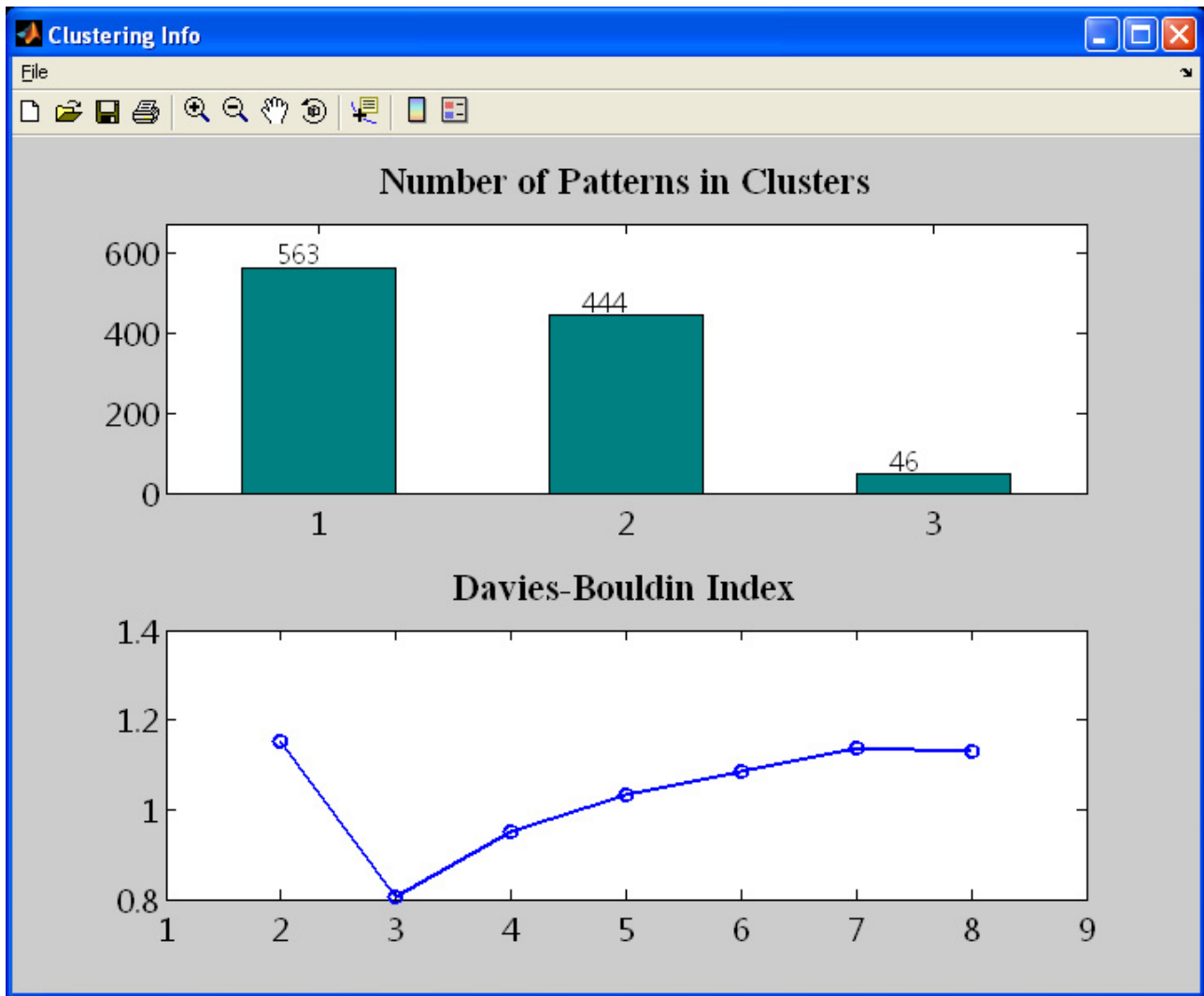


Fig. 7a. Results of K-Means clustering performed on original data From the Davies Bouldin Index a partition with three clusters is identified as most suitable

In Fig. 7a we see how many patterns were assigned to the various clusters (“**Number of Patterns in Clusters**”). Here a partition with three clusters is considered. Contrary to the nodes of the SOM (for which we claimed the property of “topological fidelity”), the cluster ID’s do not have a numerical or topological meaning. Instead of using cluster ‘1’, ‘2’, ‘3’ we could have used also ‘A’, ‘B’, ‘C’. KAnalysis, by convention, assigns a ‘1’ to the biggest cluster, ‘2’ to the second biggest and so on.

As mentioned earlier, in partitioning, non-hierarchical clustering algorithms, the number of cluster must be chosen a-priori by the user. A popular criterion for this choice is the “**Davies Bouldin Index**” (DBI) – KAnalysis offers its use as an option (see the paragraph regarding the KAnalysis configuration, i.e., “Settings”). The DBI balances the summed internal dispersion against distances measured between cluster centroids (external dispersion). In doing so we account for the trade-off between compactness of clusters (which is optimum when dispersion is minimum), and number of clusters, which is also preferred to be small. An exact definition of the DBI is given in the Appendix. The most suitable partition is identified for which the DBI is minimum. Note, that when the DBI option is used, all related results make reference to this partition.

As we shall learn later - in the context of KAnalysis configuration – Clustering can be carried out both on the original data as well as on the SOM nodes. Using the K-Means Clustering on the original data one obtains a **Clustering Info** figure such as shown in Fig. 7a. For K-Means on SOM values the **Clustering Info** figure looks like the example shown in Fig. 7b.

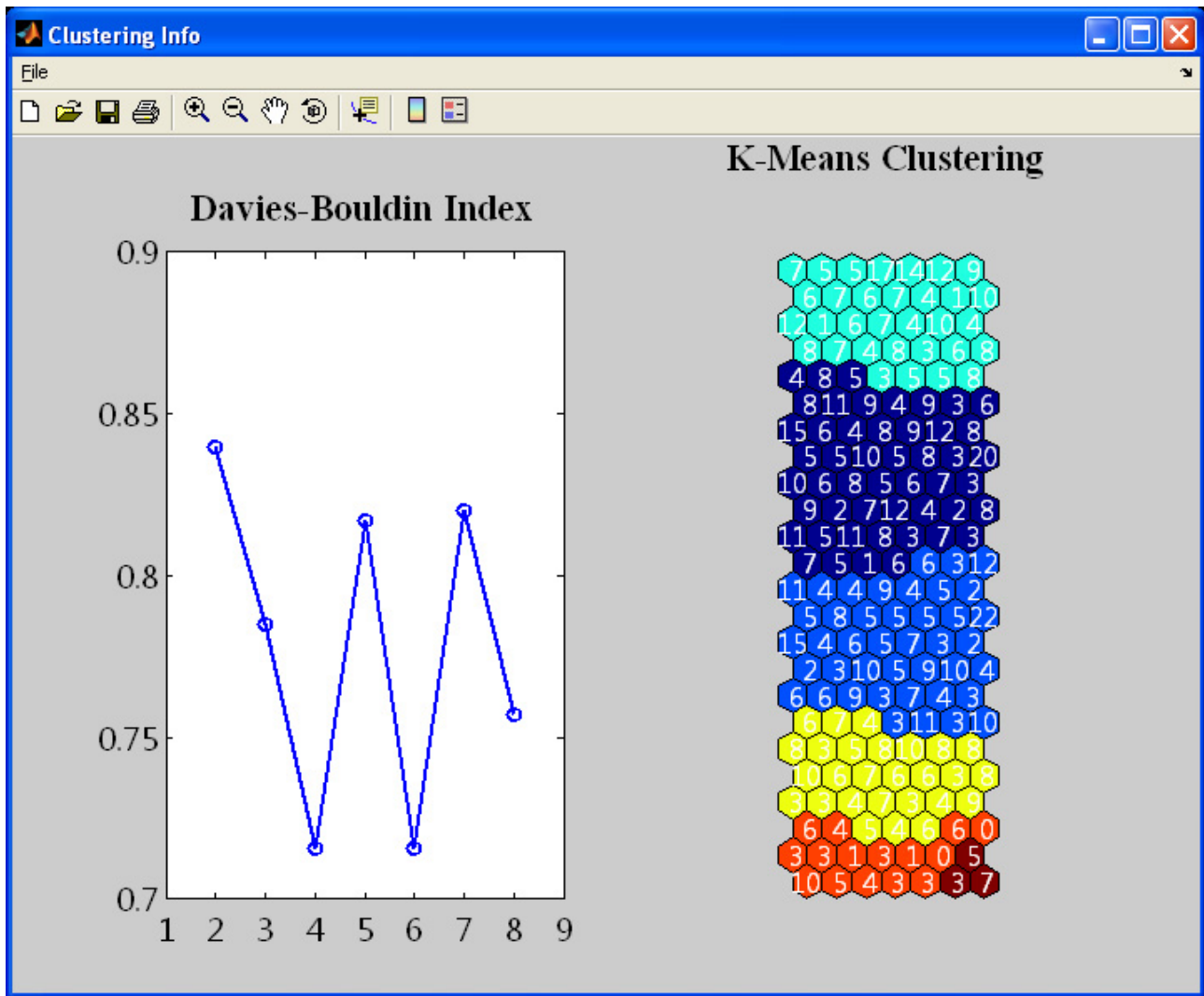


Fig. 7b. Clustering Info figure for K-Means Clustering on SOM data

Cluster ID here is given by assigning a color to the nodes of the map. Numbers given in the single nodes correspond to the number of patterns for which a node is BMU. A representation like Fig. 7b is only available for K-Means clustering on SOM data.

A further option for clustering in KAnalysis is Fuzzy C-Means. Fuzzy clustering is similar to classical, crisp K-Means clustering besides the fact that the class membership of a pattern is expressed by a vector of scores, expressing the degree to which a pattern belongs to a certain class or cluster. More about how these scores are calculated in the Appendix. The upper frame (“**Number of Patterns in Clusters**”) of Fig. 8 resembles to the one seen for the K-Means clustering, with the difference that a partition with three clusters was adopted (in fact, there’s no criterion comparable to the Davies Bouldin Index available, which may guide the user’s choice). The frame reports how patterns would be grouped considering only the highest score of class membership. The second frame “**Fuzzy C-Means Clustering Quality**” illustrates the degree of fuzziness of the clustering. In the figure we find that 18% have a maximum score of 0.9 or greater, so these clusters are very crisp. Another 49% has maximum scores between 0.7 and 0.9, which means still a rather crisp class membership, 30% reach a maximum score between 0.5 and 0.7. On the other hand, 3% have a maximum score of only 0.5 or less. These patterns are characterized by a lack of a prevailing score consequently they cannot be assigned so uniquely to a single class as the other patterns. The presence of many patterns without a prevailing score can help the user to redefine the number of

clusters used in the partition. On the other hand, one must be aware of the possibility that part of the data set may not show clear heterogeneities, but smooth transitions of features are present.

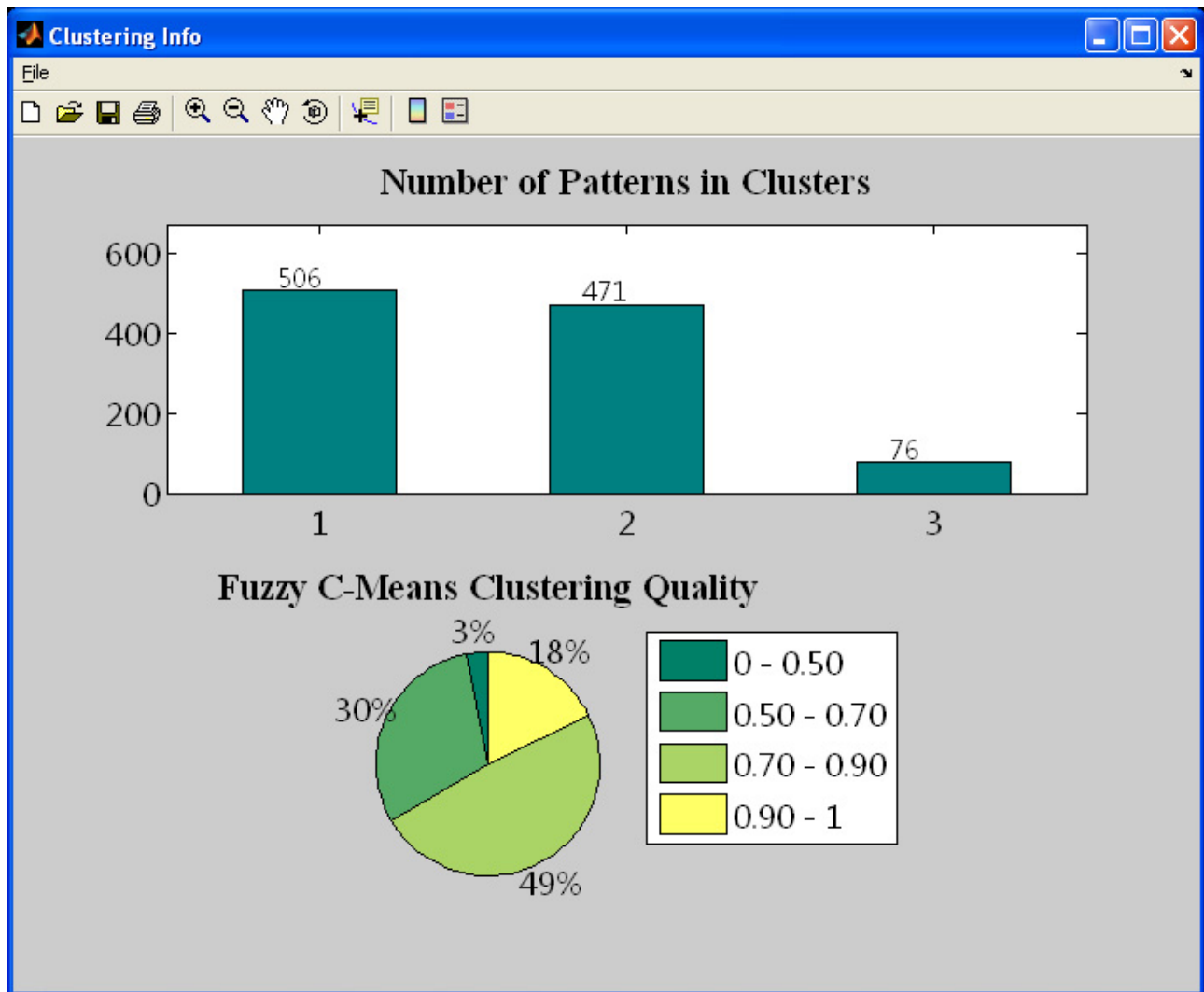


Fig. 8. Info of Fuzzy Clustering

The third clustering method provided by KKAnalysis is referred to as “CA”. It is based on the use of the adaptive determinant criterion explained earlier. Similarly to Fuzzy C-Means we have no simple rule for the identification of the “right” choice of cluster numbers. Observing Fig. 9 may help to understand whether clusters formed with this options are large (many patterns, see frame “**Number of Patterns in Clusters**”) and compact at the same time (frame “**Cluster Heterogeneity**”). For instance, the cluster #3 has 152 elements, thus being the smallest one in the partition. Its degree of heterogeneity is also the smallest one ($1.9e-2$). On the other hand, clusters #1 and #2 have somewhat larger heterogeneity ($4.0e-2$ and $3.4e-2$) but are by far prevailing. It is expected that the sum of all heterogeneities (“Dtot”) decreases when a partition with a higher number of clusters is chosen. So one tends to prefer a partition with few clusters unless noting a significant decrease of Dtot with more clusters. Note, however, that these considerations furnish only very generic guidelines whereas the final decision on the suitable partition may include other considerations going beyond these numbers.

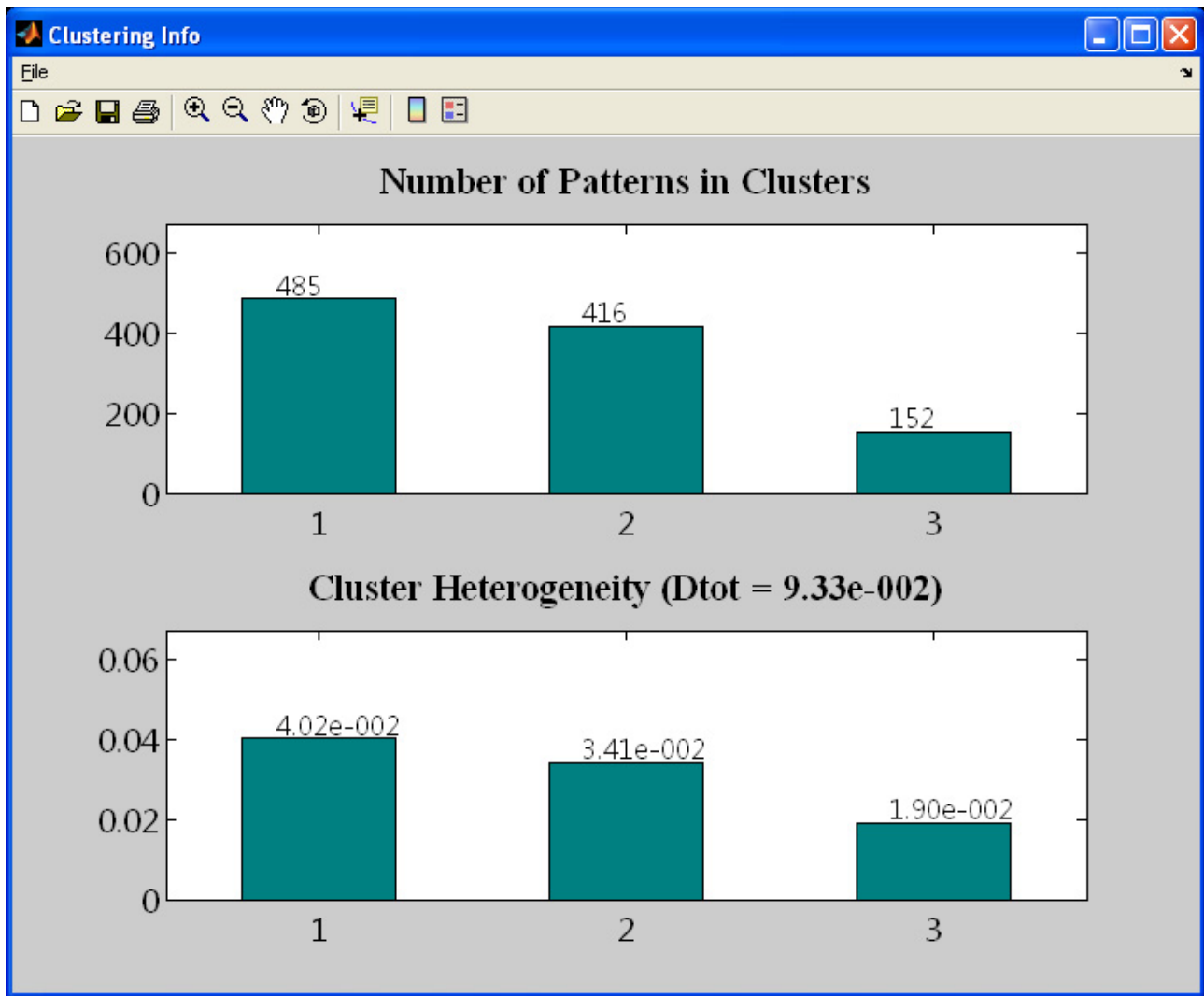


Fig. 9. Clustering Info figure for Adaptive Determinant Cluster Analysis

Note that the figure 7, 8 or 9, can be saved as MATLAB (*.fig) or TIF file and carries the name “ClustInfo_YYYYMMDDThhmmss, where YYYYMMDD is the date (year/month/day) and hhmmss is the time (hours/minutes/seconds). The “T” in the middle helps to see where the data string ends and day time information begins. Saving of graphical output is activated with the appropriate selections in the “Settings Screen”.

On the lower part of the “**Figure**” Frame (see Fig. 4c) the user has three options to visualize - pattern by pattern - the classification results obtained from the SOM and the various clustering methods. For the generation of the Fig. 10a-c we have been using cluster analysis using the adaptive determinant criterion. The corresponding figures obtained with K-Means clustering have the same layout and are therefore not shown. In the first graph, using the option “**Plot of Clusters and Kohonen ColorMap 1**”, the cluster membership of each pattern can be read from the vertical position of its colored shaped marker whereas the color of the triangle correspond to the one of the BMU the pattern belongs to.

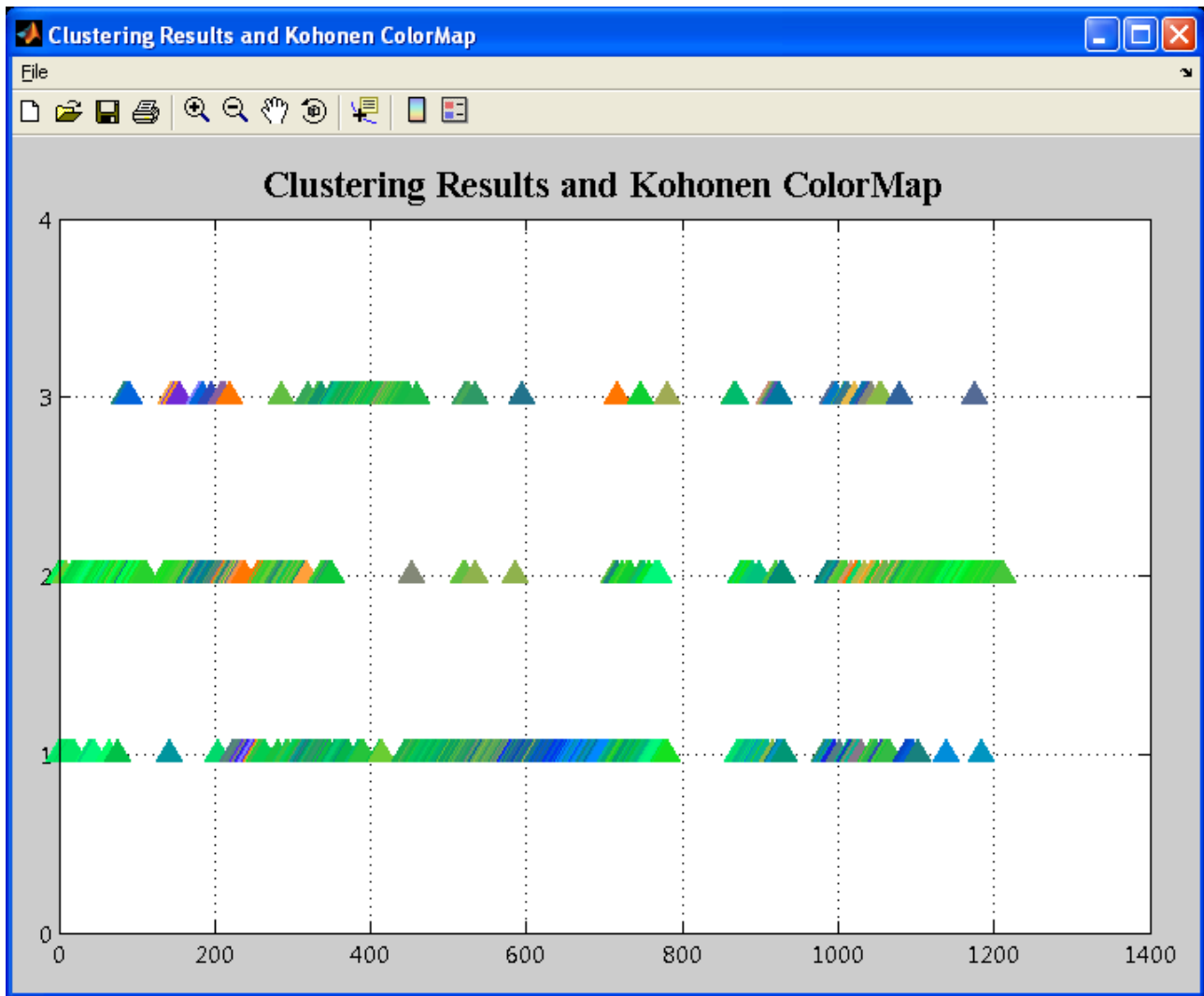


Fig. 10a. The first of three alternatives for representing the classification results (referred to as “ColorMap 1”)

In the “**Plot of Clusters and Kohonen ColorMap 2**” and “**Plot of Clusters and Kohonen ColorMap 3**” the same information is represented in an alternative manner. Choosing the former (Fig. 10b), the patterns are again depicted as triangles with colors corresponding to their BMU in the SOM, and the vertical position showing their cluster membership is now represented by grey bars. In the latter option (Fig. 10c), triangles are not at the top of grey bars but at the bottom of them, on the horizontal axis. The options “...Map 2” and “...Map 3” may be preferable when a specific column of indexes is used as “X Axis”. Jumps in the indexes, for instance caused by missing data encountered during some time intervals, become clearly evident, whereas these jumps are somewhat obscured in the first option (Fig. 10a) as the size of the triangle limits the resolution on the horizontal axis.

The option “...Map 3” may be recommended when the separation introduced by clustering is suspicious. The continuous sequence of colored triangles shown at the bottom of the graph can help to understand whether transitions from one cluster to another really correspond to sizeable changes in pattern features.

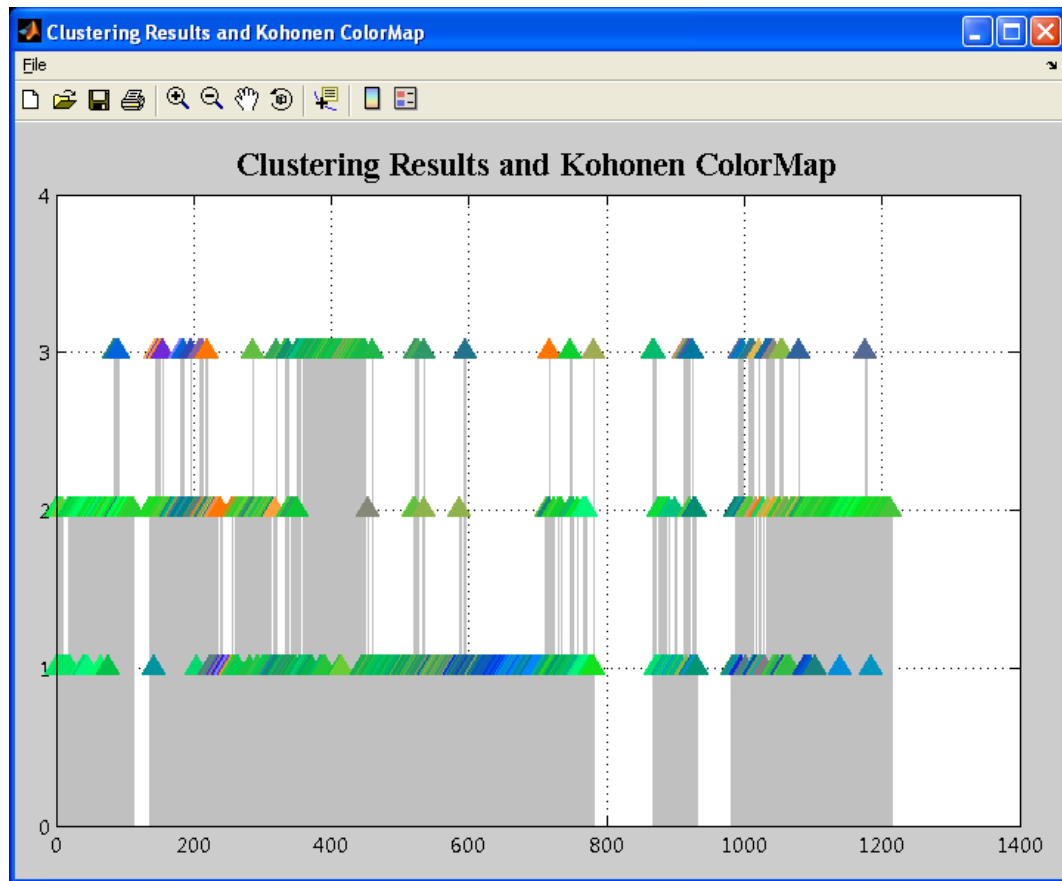


Fig. 10b. The second alternative of “Clustering Results and Kohonen ColorMap” (referred to as “ColorMap 2”)

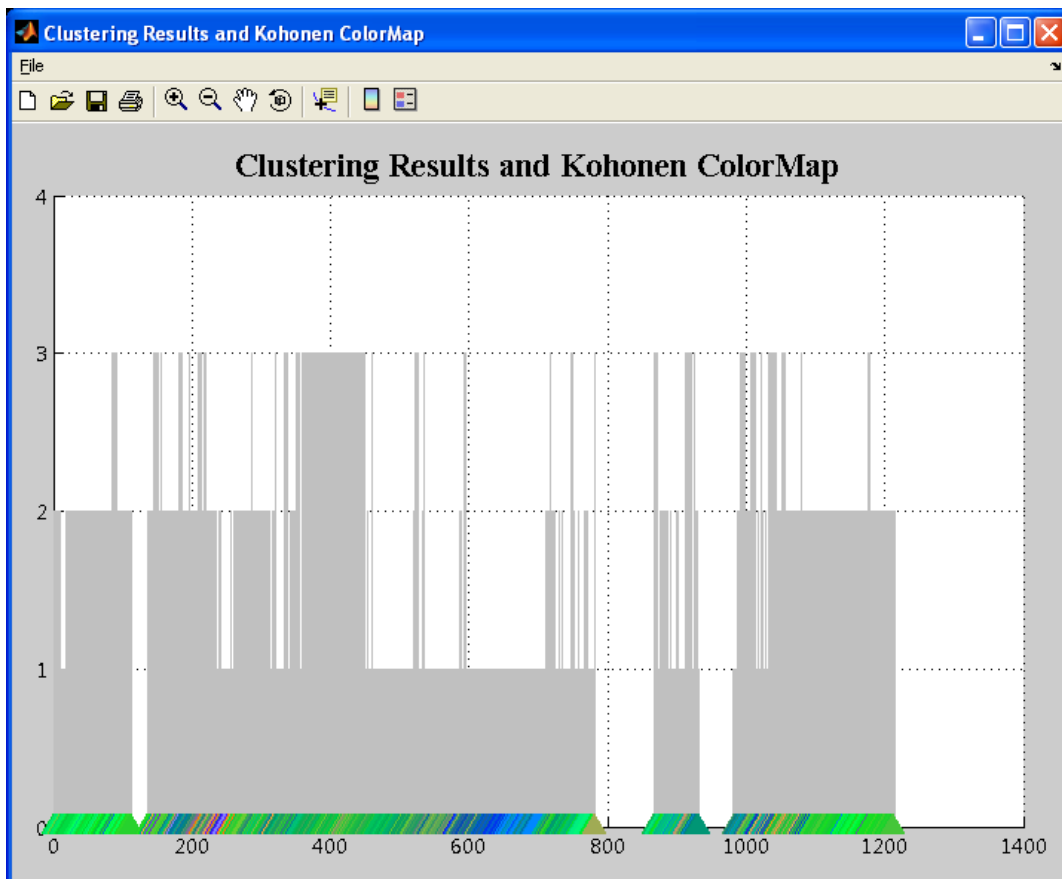


Fig. 10c. The third alternative for “Clustering Results and Kohonen ColorMap” (referred to as “ColorMap 3”)

The content of figures 10b and 10c is slightly modified when clustering is carried out with the Fuzzy C-Means option. As before the light grey bars indicate the “preferred” class membership. In addition, dark bars represent the maximum score, i.e., the maximum of all encountered class membership values for a pattern. It determines to which cluster the pattern should be preferably assigned (see Fig. 11a, b). The coverage of the light grey bar by dark grey one expresses the degree of fuzziness of the class membership. If the coverage is complete the assignment of the class membership is crisp, with no fuzziness. Fuzziness is high for patterns where a large part of the light grey bars remain visible. In the graphs, most of the patterns between #0 and #100, for instance, are assigned with a good score to the cluster 1, i.e., their membership is rather crisp. There is a certain degree of fuzziness for some patterns belonging to the group 2. For example, patterns between #660 and #710 have a maximum score of class membership around 0.7, i.e., they show a higher degree of fuzziness than the patterns between #0 and #100.

The colored stripe placed underneath the abscissa in Fig. 11a and Fig. 11b reports the full class membership vector encountered for all patterns. The cluster IDs are represented with colors. Each bar of this graph is composed of as many sub-bars as the number of clusters. Obviously, the sum of all class membership values is 1, and the length of the sub-bars is proportional to the membership rate of the patterns to the clusters. For instance, in Fig. 11a, the largest sub-bars of first 100 patterns are of blue color, consequently these patterns are preferably assigned to cluster 1.

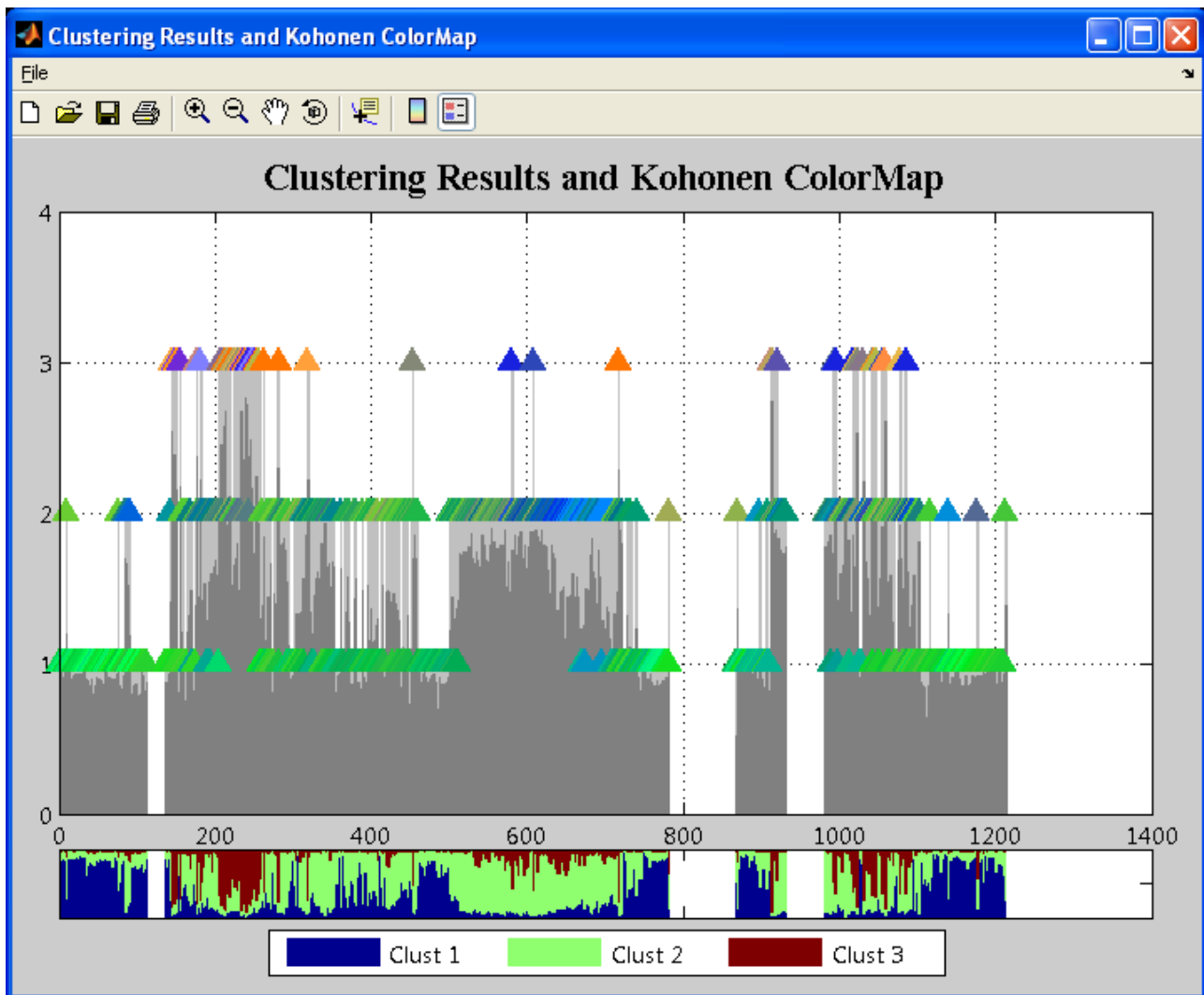


Fig. 11a. “Clustering Results and Kohonen ColorMap 2” obtained with Fuzzy C-Means clustering

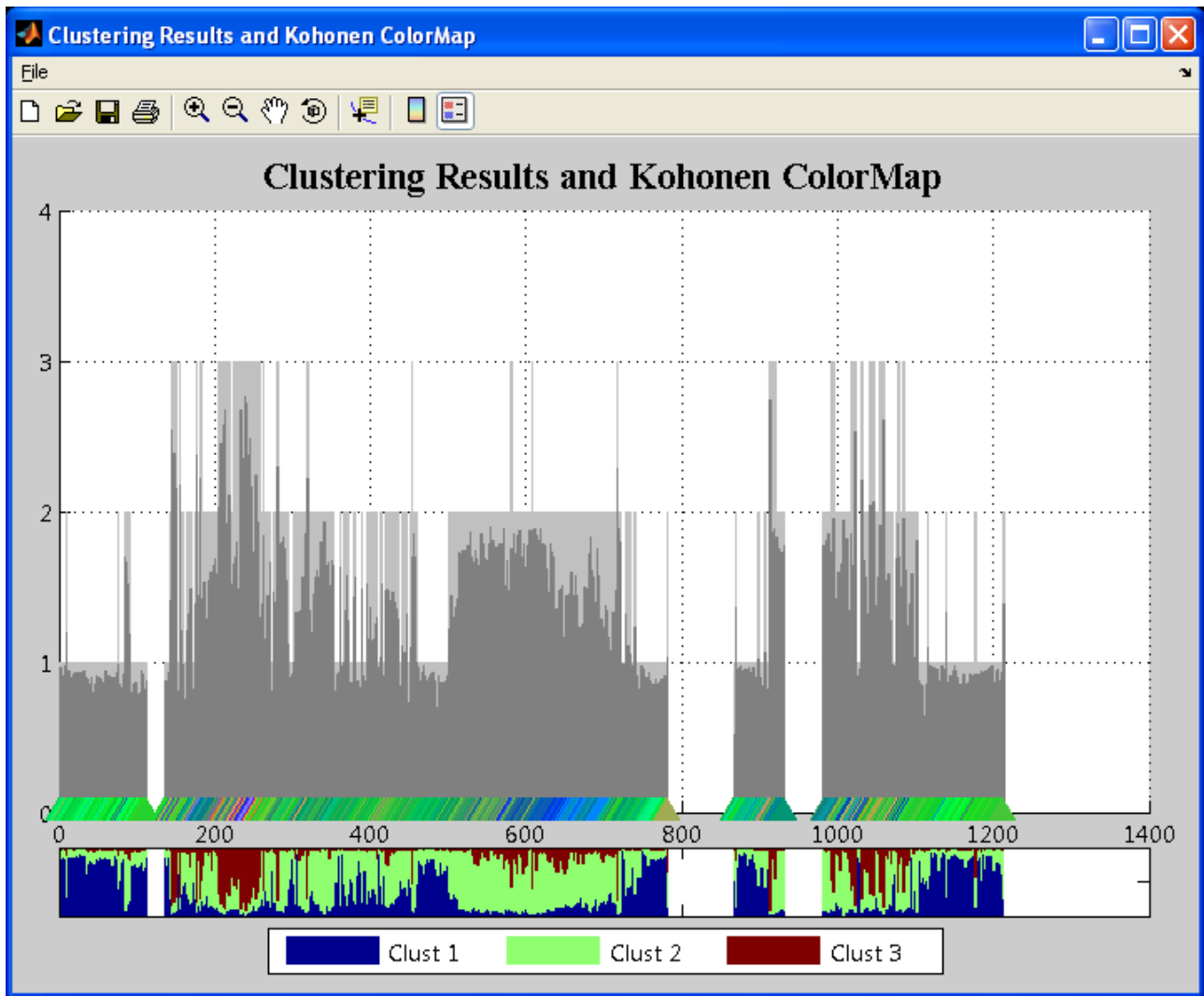


Fig. 11b. “Clustering Results and Kohonen ColorMap 3” with Fuzzy C-Means clustering

The synoptic graphs (see Figs 10 and 11) can be zoomed either by editing the plots with MATLAB or by setting the interval in the “Show” panel in the “Welcome Screen” (Fig. 4a). For creating the Fig. 12 we have applied the Fuzzy C-Means with three clusters, and specified the patterns to show as “150” for the first and “300” for the last one. In Fig. 12 the color of the BMUs and the class membership values for each pattern can be clearly recognized.

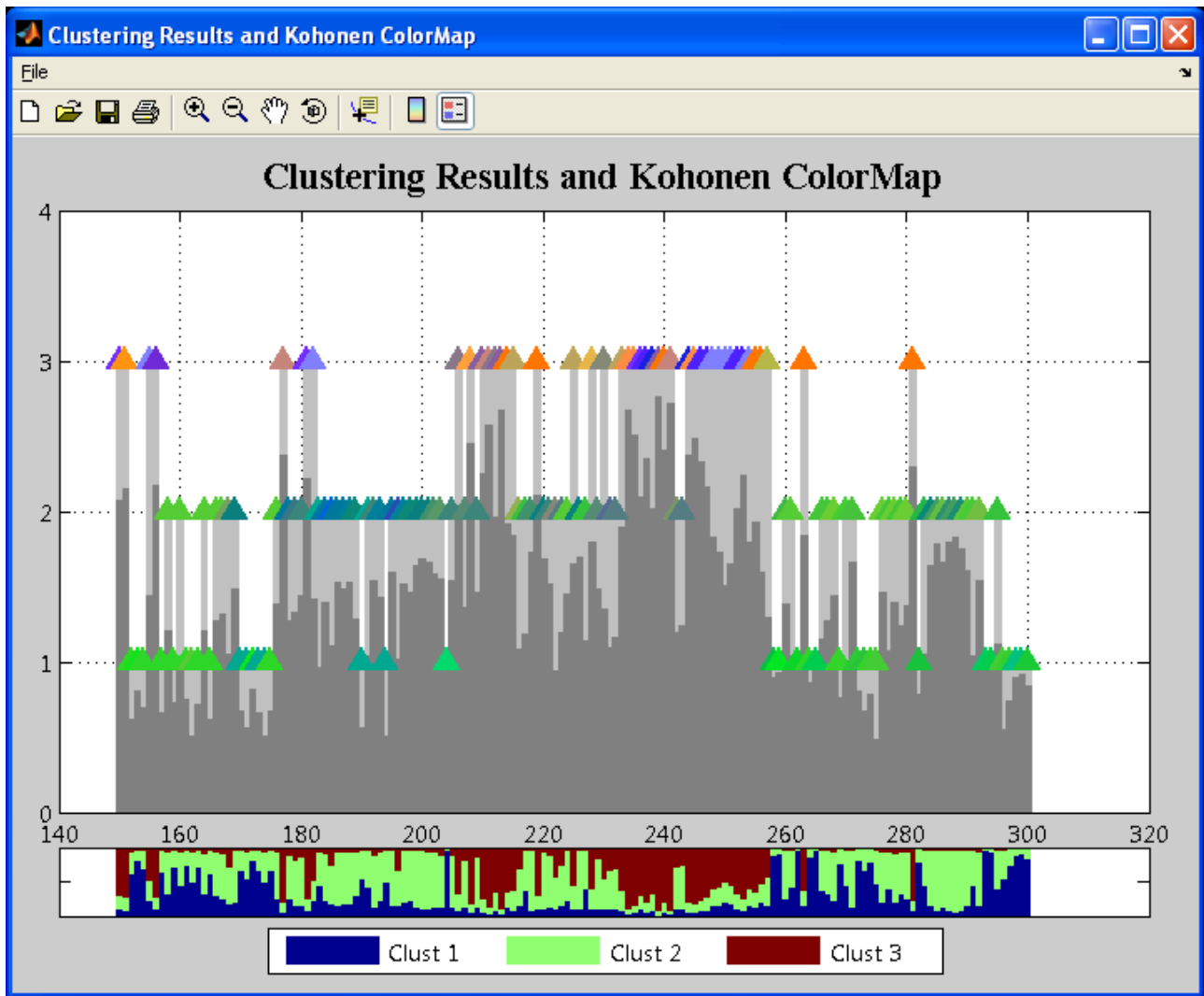


Fig. 12. An example of result using the zoom function of KKAnalysis

3 Configuring KKAnalysis – The “Settings”

KKAnalysis has been designed with the scope to be applicable to a wide variety of fields. This flexibility entails the existence of a considerable number of parameters which govern the operation of the program. Clicking the “Settings” button on the “Welcome sheet”, KKAnalysis prompts the “Settings sheet” (Fig. 13) with a number of configuration options shown below. Before discussing all the various items on the configuration sheet, please note the four buttons at the lower right corner “Load”, “Save As”, “OK” and “Cancel”. Running KKAnalysis for the first time all options are preset to their default values. After choosing the various options it is necessary to click the “OK” button in order to accept options. The program can be started subsequently. Note that changes accepted with “OK” will be valid for the current session, and are used as default in the following session.

It may be desirable to customize the configuration in order to reproduce specifically interesting runs in a later moment. For this purpose press “Save As”. KKAnalysis permits to create a configuration file, which should have an extension “*.kfg”. Clicking the “Load” button it can be loaded so that the program recovers the parameters of the desired run. Clicking “Cancel” the user stops editing the configuration and returns to the “Welcome window” ignoring all changes of the configuration, which had not been saved.

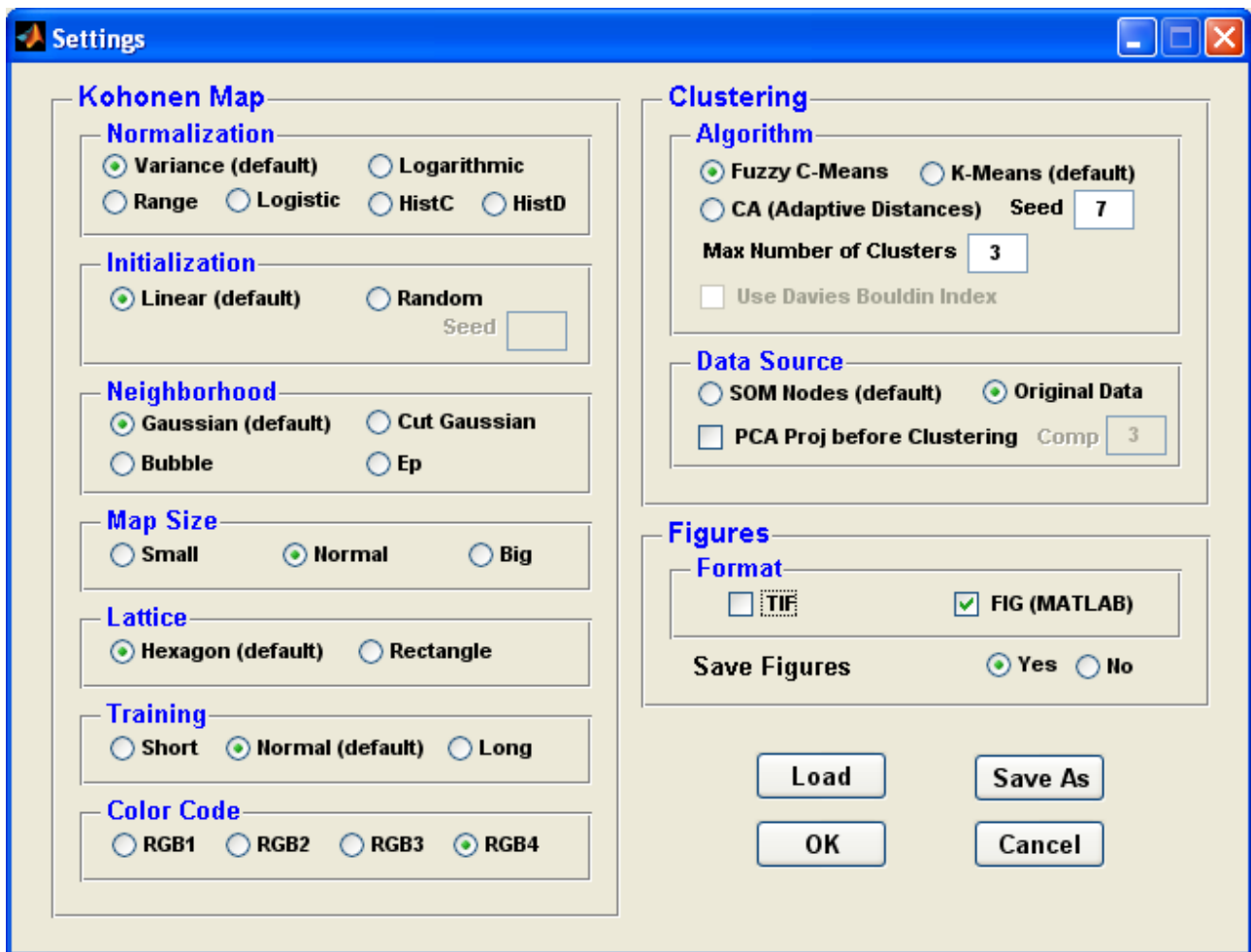


Fig. 13. The “Settings” sheet

Using the options in the “**Normalization**” frame the user has the possibility to keep the data in a certain range. KAnalysis uses in this context the procedures provided by the SOM Toolbox 2. All normalization operations are carried out considering each component (feature) separately. In detail: “Variance” is the default option and means that, after subtracting the average - $mean(x)$ - of all x , the normalization is carried out with respect to standard deviation - $stdev(x)$. The normalized values x' are obtained with the relation : $x' = (x - mean(x)) / stdev(x)$.

“Logarithmic” is the preferred option for data with a high dynamic range. Normalization is carried out using the formula $x' = \log(x - min(x) + 1)$, $min(x)$ being the minimum of all values in the data set.

“Range” scales values so that x' ranges from 0 to 1, i.e., $x' = (x - min(x)) / (max(x) - min(x))$, $max(x)$ being the largest value in the data set.

“Logistic” uses the famous sigmoidal function: $x' = 1 / (1 + exp(-x'))$, x' being the normalized value of x with respect to the standard deviation, just like in the “Variance” normalization.

“HistD” and “HistC” transform the metric values of x into ordinal ones, sorting them with respect to their amount and keeping only the index (or rank). In “HistD” sorting and ranking is carried out as is, whereas in “HistC” the ranking is carried with respect to bins, which is more effective with respect to sorting large data sets. Ideally the bins contain the same number of patterns. As in “HistC” the number of bins is obtained from the uprounded square root of the number of patterns, consequently the distribution of data within the bins will not be perfectly uniform. The data falling within a certain bin are linearly scaled with respect to the lower and upper rank boundary of the bin. For example, data in the bin 3 will be linearly transformed such that they cover the range between 3 and 4.

The “**Initialization**” frame has two options, “linear” and “random”.

The linear option is based on the eigenvalues of the covariance matrix. Having a bi-dimensional SOM only the two largest eigenvalues are considered. The initial weight of each node of the bi-dimensional SOM is then obtained interpolating along the two largest eigenvectors, which represent length and width of the SOM.

The “random” option can be useful if the calculation of the eigenvalues of the covariance matrix is ill-conditioned. In this case the initial values x_i for the i -th feature is set to $rand(1)*(max(x_i)-min(x_i))$.

The “**Neighborhood**” frame offer various choices how the influence area during SOM training is defined and how updating of nodes neighboring a BMU is handled. It was mentioned earlier that the distance dependence of the upgrade of a node is described by a function φ . This function can be “Gaussian” (this implies that all nodes are considered even having small and gradually decreasing importance), “Cut Gaussian” (i.e., tails of the Gaussian representing distant nodes are omitted), “Bubble” (a rectangular or box-car function, 1 inside the radius of influence, 0 outside), “Ep” (importance decreases proportionally to the square of the distance).

An important issue is the “**Map Size**”. KAnalysis follows the strategy applied in the SOM toolbox 2, where the design of the map is based on the covariance matrix of the data and its eigenvectors. The number of nodes for a “Normal” sized map is obtained heuristically according to $n_nodes = 5 * n_patterns^{0.54321}$. Then, the ratio of the two largest eigenvalues is used to determine the length and the width of the map. The actual side lengths are chosen in a way that the resulting number of nodes is as close as possible to the number of nodes calculated with the heuristic formula. The choice “Large” gives a map with doubled side length (i.e., 4 times the number of nodes of “Normal”), conversely “Small” produces a map where the side length are only 50% of a “Normal” sized map.

The “**Lattice**” can be made up either by “Hexagon(s)” or “Rectangle(s)” nodes. The choice affects the definition of neighborhood of nodes. In a hexagon lattice each node has 6 neighbors with exactly the same distance, in a rectangle lattice only four. From a geometrical point of view hexagons can be considered as some kind of optimum polygon. With hexagons one can cover an area without leaving gaps, minimizing the sum of side length. The “Hexagon” option is proposed as default choice.

The “**Training**” is carried out using the batch training algorithm proposed in the SOM Toolbox 2. Batch training means that an upgrade of the node weights is made only once during a cycle, i. e., after comparing all patterns with the nodes of the map. Conversely, in sequential training - not adopted here - weight are updated every time a pattern is presented to the map. Benchmark tests revealed that batch training is the fastest one among other methods. The training length depends basically on the ratio of the number of data and nodes of the SOM, referred to as mpd . Suppose a map with 100 nodes and 450 patterns in input, then $mpd = 0.22$. KAnalysis then carries out $ceil(10 * mpd) = ceil(2.2) = 3$ steps in ‘rough tuning’, and $ceil(40 * mpd) = 9$ steps in ‘fine tuning’ mode, adjusting the learning rate first rapidly, then more slowly. Choosing “short” training, the number of steps is four times smaller, for “long” training 4 times larger. The learning rate is adjusts as a function of time following a relation $\lambda(t) = \lambda_0 / (1 + 100t/T)$, T being the training length and λ_0 the initial learning rate. It is of 0.5 in the “rough” tuning phase, and 0.05 in the “fine” tuning phase.

The **Color Code** option allows the user to choose among different configurations of heuristic color codings. It permits to show the same results with different color maps. It can be possible that some details are visible in a better way using a color code rather than another one. RGB1 is the default. The first eigenvalue and eigenvector is assigned to “Red”, the second to “Green”, whereas “Blue”

corresponds to a linear combination of the first two eigenvalues. In the options RGB2, RGB3, RGB4 the assignment of colours to eigenvalues and –vectors is changed in a robin-around manner. For the example shown in this document, we have been using the “RGB4” option.

In the frame “**Clustering Algorithm**” the user may choose among the classical “K-Means”, the “CA(Adaptive Distances)”, which uses the adaptive determinant criterion explained earlier, and the “Fuzzy C-Means” option. All algorithm are partitioning ones, so the number has to be specified a-priori inserting the desired number in the field “Maximum Number of Clusters”. Note, that the term “Maximum” is related to the K-Means option. In this case, setting the tick in the field “Use Davies – Bouldin Index” clustering is carried out for a cluster number varying from 2 to the number given in the field “Maximum Number of Clusters”. Finally the results are reported which correspond to the optimum partition inferred using the “Davis Bouldin Index”. On the contrary, for any option, where the Davies-Bouldin Index is not used, KAnalysis creates only the partition corresponding to the number of clusters selected in the field “Maximum Number of Clusters”.

In “**Clustering Source**” the user specifies whether clustering is carried out on the raw data or using the SOM values. With the latter option the scatter of patterns considered in clustering may considerably reduce, which can facilitate the task. Note with the adaptive distance option there is a risk of ill-conditioned inverse of the dispersion matrix, especially when the number of components is high and only a limited amount of patterns is available.

It is possible to simplify the clustering problem reducing the dimensionality of data performing a Principal Component Analysis. For this option set a tick on “PCA Proj. before clustering” and specify the number of principal components to be considered (field “Comp”). Of course, the number of principal components should be less or equal the original number of components of the data set or SOM.

In the frame “**Figures**” the user can choose between the possibility of saving the produced figures or not. In the former case, there are two available formats: TIF format that guarantees an optimum quality of the picture and FIG format that allows the user to open and edit the picture using MATLAB. Note that figure files saved in the MATLAB format are much smaller than corresponding TIF images.

Appendix

SOM Quality

The issue of SOM quality is a complicated one. Typically two evaluation criteria are used: resolution and topology preservation. If the dimension of the data set is higher than the dimension of the map grid, these usually become contradictory goals. The first value returned concerns the resolution and is addressed to as quantization error QE . In the introduction we learned that SOMs are created on the base of the Euclidean distance between map nodes and the feature vectors, i.e.,

$$D_{ij} = \sqrt{(\mathbf{W}_i - \mathbf{V}_j)^T (\mathbf{W}_i - \mathbf{V}_j)}$$

Once concluded the training phase, we define a generalized measure summing the distances between the feature vectors and the BMUs they belong to:

$$QE = \sum_k \sum_i D_{ik}$$

with k running from 1 to the number of BMUs and i from 1 to the number of patterns belonging to the k -th BMU. QE is the quantization error of the BMU and decreases with the number of BMUs.

Topological Error

A fundamental clue of SOMs resides in the fact that units having the smallest distances between each other are ideally placed side a side. In terms of map representation we may express the neighborhood relation on the map as some kind of “topological distance” (TD). These distances are calculated along the map grid. Consider, for example, the case of a 4x3 map. The unit ('l' to 'c') positions for 'rect' and 'hexa' lattice (and 'sheet' shape) are depicted below:

'rect' lattice	'hexa' lattice
-----	-----
1 5 9	1 5 9
2 6 a	2 6 a
3 7 b	3 7 b
4 8 c	4 8 c

Most neighboring nodes ideally have a topological distance of 1, such as the elements labeled with **2, 5, a, 7** have a topological distance of 1 with respect to element 6, whereas the elements 1,9,b,8, 3 have the TD=2. Suppose now, that during training node labeled '9' is found to be closer to the node with '6' than one of the nodes **2, 5, 7, or a**. Then, in the “rect” lattice there will be a node closest to another, whose “topological” distance is 2 instead of 1. This is undesired and consequently is considered as a topological error. On the other hand, in the “hexa” lattice the element **9** has a TD =1, so no topological error is encountered in this case.

The global topological error reports the number of cases where a $TD > 1$ occurs even though two elements are less distant from each other. A high global topological error is a diagnostics of an unsuitable choice of the map geometry. For instance, the map representations proposed here (i.e. 'sheet' shape) may fail for patterns distributed on a circle or a sphere.

The Davies Bouldin Index

The Davies Bouldin Index (DBI) (Davis and Bouldin, 1979, see also Stein et al., 2003) provides a simple guideline for the choice of a suitable number of clusters in partitioning clustering algorithms. The DBI for the partition is obtained by comparing the average similarity encountered among all clusters to the largest one.

The similarity between clusters i and j are given by

$$R_{ij} = (s_i + s_j) / \| \mathbf{c}_i - \mathbf{c}_j \|,$$

where s_i and s_j are the variances measures in each cluster, and $\mathbf{c}_i, \mathbf{c}_j$ are the corresponding cluster centroid vectors. With $R_i = \max (R_{ij})$ the DBI is obtained from the average of the R_i taken over all k clusters, i.e.,

$$\text{DBI} = 1/k \sum_i R_i$$

Remark: We desire clusters with low scattering, and a high distance among each other. Thus a minimization of the DBI is desired. The definition of cluster similarity used in Davies Bouldin Index is based on a the concept of spherical clusters, i.e., with patterns scattering more or less isotropically around a centroid. We therefore decided to use the Davies Bouldin Index only in K-mean clustering.

Fuzzy C-Means clustering

A principal difference between fuzzy and hard clustering relies in the fact that in the latter a pattern is assigned exclusively to one class, whereas in the former each pattern may belong to a certain degree to all possible classes. In fuzzy clustering, the class membership of a pattern, rather than being a simple ID, is given by a vector. The partition is consequently described by a $M \times K$ matrix of class member ship values, with M being the number of patterns in the entire data set, and K the number of clusters.

The optimum partition is determined minimizing the so called “Objective Function”

$$J = \sum_i J_i$$

$$J_i = \sum_k m_{ik}^q d_{ik}^2$$

where m_{ik} is the membership value of pattern i with respect to cluster k , d_{ik} its Euclidean distance from the cluster centroid, and q a weighting exponent (set to $q=2$ in our application). Note that the cluster centroid is not any longer given as a mere average vector of patterns belonging to a cluster, but are obtained following the relation

$$\mathbf{c}_i = \sum_k m_{ik}^q \mathbf{u}_k / m_{ik}^q$$

\mathbf{u}_k being the k -th pattern vector.

References

- Anderberg, M.R., 1973. Cluster analysis for applications, Academic Press, New York, 359 pp.
- Davies, D.L., Bouldin, D.W., (1979). A cluster separation measure. IEEE Trans. IEEE Transactions on Pattern Analysis and Machine Learning, 1(2), 1979. 224-227.
- Duda, O.D., Hart, P.E., Stork, D.G. (2001). Pattern Classification. John Wiley and Sons, 654 pp.
- Kohonen, T., 1984. Self-organization and associative memory, first ed. Springer Series in Information Sciences, vol. 8, Springer-Verlag, New York.
- Späth, H., 1983, 1985. Cluster-Formation und Analyse, Theorie, FORTRAN-Programme, Beispiele. Oldenbourg, München. Cluster-Formation und -Analyse. Theorie, FORTRAN-Programme, Beispiele. R. Oldenbourg-Verlag, München 1983. English Translation: Cluster Dissection and Analysis. Horwood, Chichester 1985.
- Stein, B., Meyer zu Eissen, S., Wißbrock, F. (2003). 3rd IASTED Int. Conference on Artificial Intelligence and Applications (AIA 03). Benalmádena, Spain, September 2003. Edited by M. H. Hanza. pp. 216-221, ISBN 0-88986-390-3.
- Vesanto, J., Himberg, J., Alhoniemi, E. & Parhankangas, J., 2000. SOM Toolbox for Matalab 5. Report A57, <http://www.cis.hut.fi/projects/somtoolbox>.