CLASSE – Synthetic Classification for Industrial Supervision

VISRED

Data Visualization by Space Reduction

V1.2r

User Guide

António Dourado
Edgar Ferreira
Paulo Barbeiro

CISUC/FCTUC

20 April 2007

TABLE OF CONTENTS

Page

TABLE OF CONTENTS 2
OVERVIEW 4
OVERVIEW 4
Description: 4
DATA IMPORT 6
- EXCEL FILE READING 6

NORMALIZATION 7
- NORMALIZATION OF DATA 7
  - Off 7
  - Mean Value Only 7
  - Standard Deviation Only 7
  - On 7
- PLOT AND BOXPLOT BUTTONS 8

DIMENSION REDUCTION 8
- DISSIMILARITY METRIC 8
- DIMENSIONS REDUCTION 8
  - Classical Multi Dimensional Scaling (CMD) 9
  - Multi Dimensional Scaling (MDS) 9
  - Principal Component Analysis (PCA) 10
- PLOTING DATA 10
- SAVE AND LOAD REDUCED MATRIXES 10
- READING FROM A MAT FILE 11

EIGEN VALUES BASED ANALYSIS 11
- EIGEN VALUES BASED ANALYSIS 11

NON LINEAR PRINCIPAL COMPONENT ANALISYS 12
- NON LINEAR PCA 12
- CLUSTERING ALGORITHM 14
  1. Hierarchical Clustering 15
  2. K-Means Clustering 15
  3. Fuzzy C-Means Clustering 16
  4. Fuzzy Subtractive Clustering 16
  5. Dignet Clustering 17
  6. SOM Clustering 17
- CLUSTERING METHOD COMPUTE 18

SAVING RESULTS 19
- SAVE ON MAT-FILE 19
- SAVE ON EXCEL FILE 19
- SEND TO *WORKSPACE* 20
- LOAD CLUSTERED DATA 20

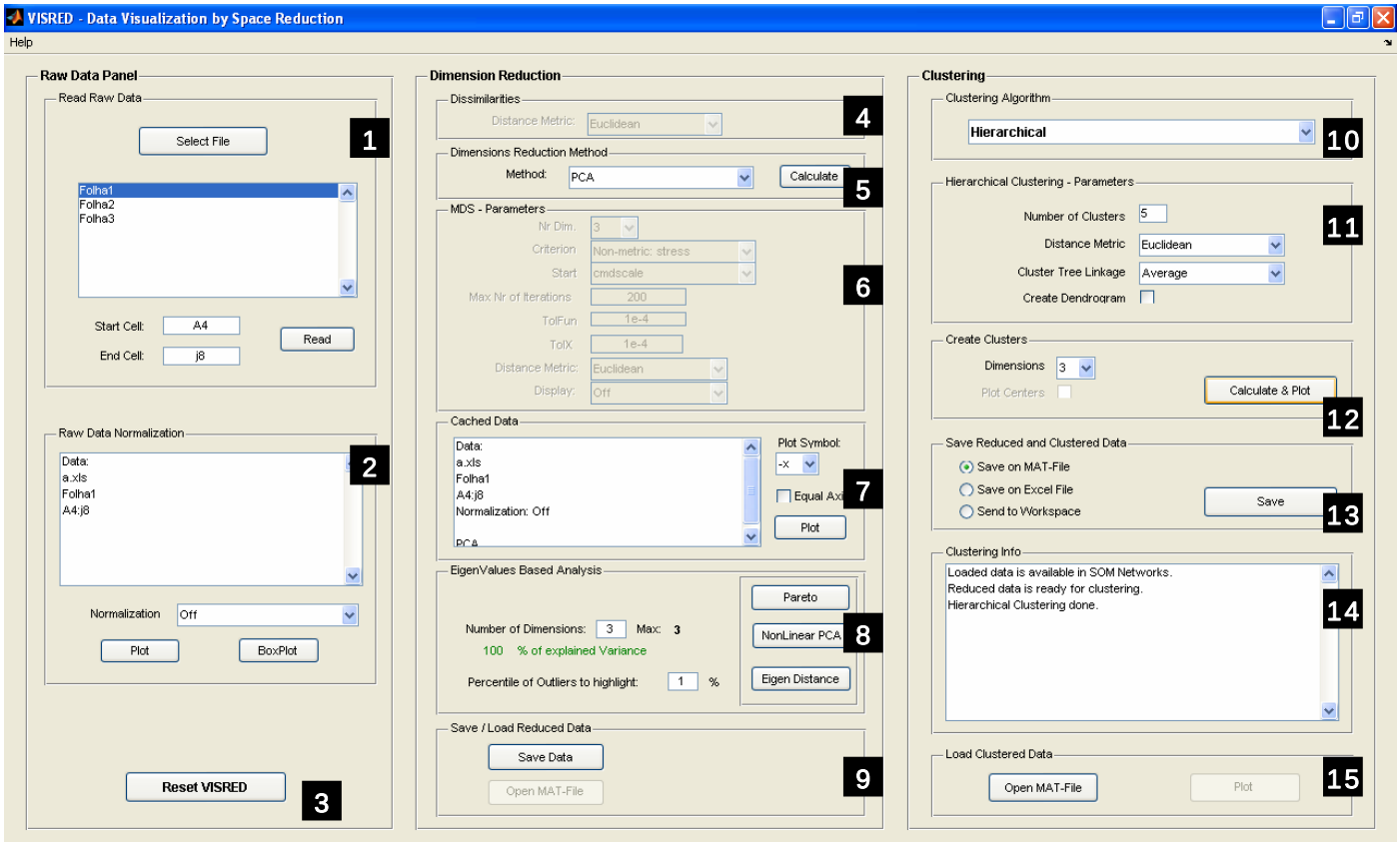
SOM CLUSTERING INTERFACE 21
- SOM INITIALIZATION 21
- SOM CLUSTERING 22
  1. SOM K-Means 22
  2. SOM U-Matrix 22
  3. Hierarchical Clustering 22
  4. Fuzzy Subtractive Clustering 22
- SAVE SOM RESULTS 23
- LOAD SOM STRUCTURE 23

RECURSIVE SOM INTERFACE 24
- STUDY METHOD 24
- RECURSIVE SOM TRAIN 25
- RECURSIVE SOM SIMULATION 25
- SAVE RECURSIVE SOM RESULTS 25

Authors:
Edgar da Silva Ferreira – edgars@student.dei.uc.pt
Paulo Filipe Domingues Barbeiro – barbeiro@student.dei.uc.pt

Supervisor:
António Dourado Correia (PhD) – dourado@eden.dei.uc.pt

OVERVIEW



Description:

1. *Read Raw Data*: Reads data from an excel file
2. *Raw Data Normalization*: Allows to normalize and visualize imported Data
3. *Reset VISRED*: Restarts all components and variables
4. *Dissimilarities*: Method to calculate dissimilarities
5. *Variable Reduction Method*: Method to reduce data from dissimilarities matrix
6. *MDS Parameters*: MultiDimensionalScaling parameters selection
7. *Cached Data*: Panel with information about data currently being analysed
8. *EigenValues based Analysis*: Several methods of visualization and optimization based on existing eigenValues
9. *Load /Save Reduced Matrix*: Reads a file containing a already reduced matrix, saved by this application
10. *Clustering Algorithm*: panel that allows choosing clustering method
11. *Clustering – Parameters*: panel that allows user to change clustering parameters
12. *Create Clusters*: panel that allows user to create clusters
13. *Save Reduced and Clustered Data*: panel that allows user to save results to an Excel or MAT file, or to Matlab *Workspace*
14. *Clustering Info*: panel that displays clustering related information

15. *Load Clustered Data*: panel containing buttons that allow user to load and plot clustered data

Tip: plots are made on separated windows.

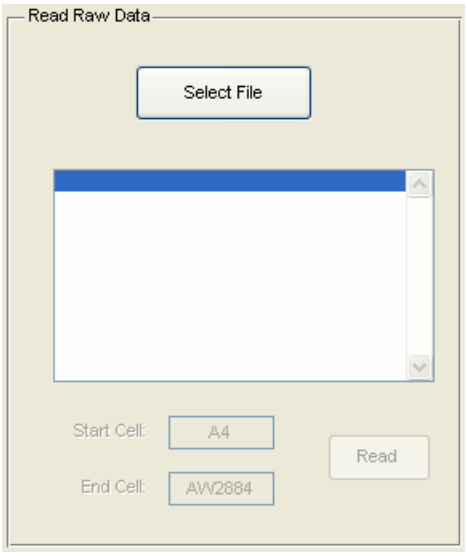
DATA IMPORT

Excel file reading

The data that the user intends to subject to dimension reduction must be read in a sheet from an Excel file. In this case, the first step to make in the application is importing the data, in the *Read Raw Data* panel.

For such, user must choose a file by pressing *Select File* button.

After this, all Excel sheets in the file are listed below, and user only have to choose which sheet contains the data that he wants to analyze.



By default, values that appear in the cells refer to data with 47 variables, which are the study target in this project.

The initial cell (left top corner of data) corresponds to the variable names line (if exists, else is the first line of data), and to the first column to the labels of each event (if there are no event labels, first column of data). The end cell is the right bottom cell with data. To read the data, user must press the *Read* button.

Multi labels for headers, variables and events are accepted. Also, if there are no labels at all (or labels are incomplete), automatic ones will be generated.

Header1 1 (1)	Header 2 (1)	…	Header m (1)	Variable1	Variable2	…	VariableM
Header1 1 (2)	Header 2 (2)	…	Header m (2)				
…	…	…	…				
Header 1 (N)	Header 2 (N)	…	Header m (N)				
1st eventLabel_1	1st eventLabel_2	…	1st eventLabel_m	11	12	13	14
2nd eventLabel_1	2nd eventLabel_2	…	2nd eventLabel_m	21	22	23	24
3rd eventLabel_1	3rd eventLabel_2	…	3rd eventLabel_m	31	32	33	34
4th eventLabel_1	4th eventLabel_2	…	4th eventLabel_m	41	42	43	44
…	…	…	…	51	52	53	54
…	…	…	…	61	62	63	64
nth eventLabel_1	nth eventLabel_2	…	nth eventLabel_m	71	72	73	74

If there are several Variable Rows for headers and Variables Labels, they will all be concatenated to just one row.

Note:

All labels **have to be text cells**, or else Matlab will recognize them as numeric values and attach them to the data.

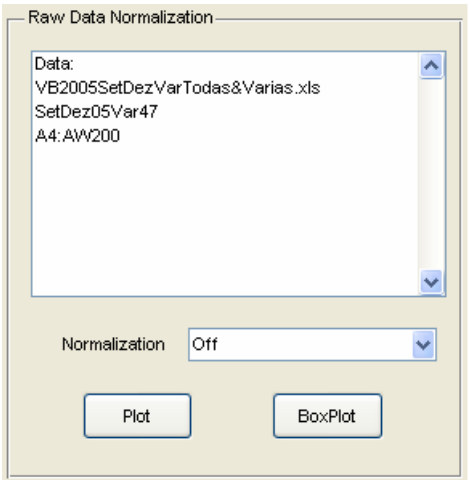
Also, if data is incomplete (numeric cells without values), errors will appear in the attempt to calculate the dissimilarities matrix and respective dimension reduction.

NORMALIZATION

Normalization of Data

Generally speaking, collected data for data-mining processes regards to variables with different measurement units (or even with no unit). The absence of homogeneity on the data scale can induce errors on the analyzing process. To prevent this from happen, normalization is required, although some important information can be lost during this process.

The method of normalization is chosen in the *Raw Data Normalization* panel



- *Normalization:* Can be one of four values

  - Off
    Data is used as read

  - Mean Value Only
    Calculates and subtracts to each variable their mean value. By doing so, the mean of each variable becomes zero.

  - Standard Deviation Only
    This method sets the maximum standard deviation (STD) of each variable to be one.
    The regular STD based normalization set all STDs to 1, which in cases of constant (or near constant) variables results in a division by zero (or near zero). When this happens, those variables are amplified to ∞, being then much more influent than the others, and unbalancing the proportion between data.
    By allowing variables to have STD≤1, this situation is corrected – those who have already a low STD remain unchanged.

  - On
    This option is equivalent to '*Mean Value Only*' and '*Standard Deviation Only*', and the resulting data has mean = 0, and STD≤1

Plot and Boxplot Buttons

Boxplot button– Draws a box and whisker plot with one box for each variable

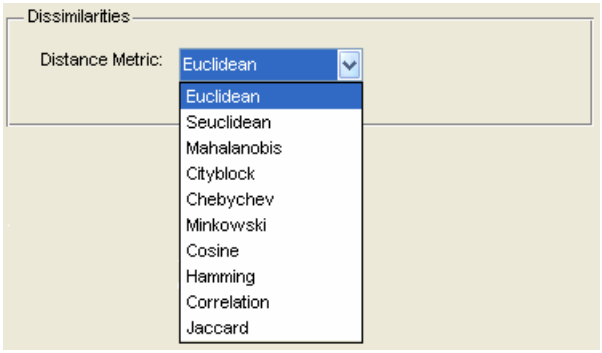
Plot button– Plots read data, with events on X axis and Variable Values on Y axis. Clicking on points of lines will show to which variable and event is selected.

This point selection option to see information about points is available throughout the application, and can be enable or disable by pressing the data Cursor Button  at anytime. When active, the mouse pointer will became a small white cross ╪. Keyboard arrows can be used to cycle along the data points, and by pressing Alt key while selecting a new point with the mouse, multiple tags will appear.

DIMENSION REDUCTION

Dissimilarity metric

Dissimilarity is a measurement of disparity between several multidimensional points. There are different methods for calculating distances, being all of them available in the interface.



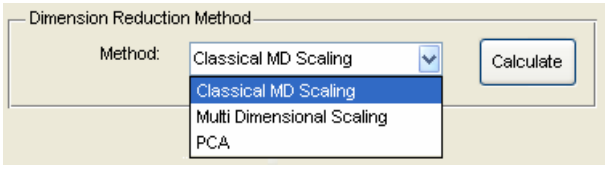
Note: From our experience, best results are obtained with Euclidean, SEuclidean, CityBlock and Correlation metrics.

(For further information, see *pdist* in Matlab Help)

Dimensions Reduction

Note: Before starting calculus it may be useful to activate CPU monitoring, since that, depending on the amount of information, it may take from 10 seconds up to 2 days for calculus to be complete. CPU monitoring can be activated by pressing *Ctrl+Alt+Del*, and by minimizing the console that appears.

The method of dimensions Reduction can be choose on the *Dimension Reduction Method* panel.



- Classical Multi Dimensional Scaling (CMD)

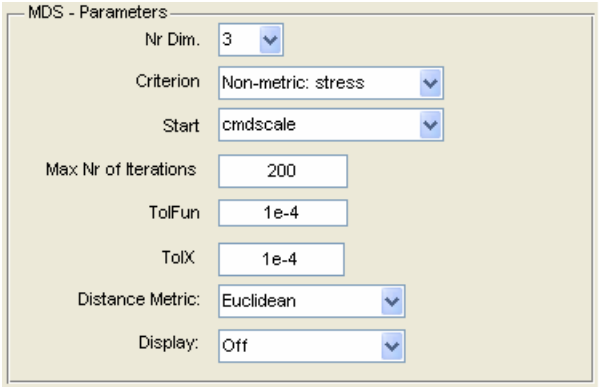
  This method takes an interpoint distance matrix, and returns a configuration matrix in a smaller dimensional space.

  Another output is a eigen values vector, where each eigen value corresponds to the representativiness of the each new dimension.

  (For further information, see *cmdscale* in Matlab Help)

- Multi Dimensional Scaling (MDS)

  Multidimensional scaling is an optimization method for dimensions reduction that tries to represent data on a fixed number of dimensions, by equalizing the dissimilarities maps of both matrixes.



  There are several parameters that can be chosen to optimize performance, and below is made a short description of them.

Parameters:

Nr of Dimensions: Desired number of dimensions for data after reducing process.

Criterion: The goodness-of-fit criterion to minimize. This also determines the type of scaling, either non-metric or metric, that *mdscale* performs.

Start: Initial configuration of points. *Cmdscale* is suggested due to its better performance.

Max Number of Iterations: Maximum number of iterations allowed.

TolFun: Termination tolerance for the stress criterion and its gradient.

TolX: Termination tolerance for the configuration location step size

Distance Metric: Metric for calculation of dissimilarities during the attempt to minimize criterion. For coherence, one should use the same metric on dissimilarities and distance, although it is possible to try combinations of metrics.

Display: Allows to define the amount of displayed output on Matlab's console.

(For further information, see *mdscale* in Matlab Help)

- Principal Component Analysis (PCA)

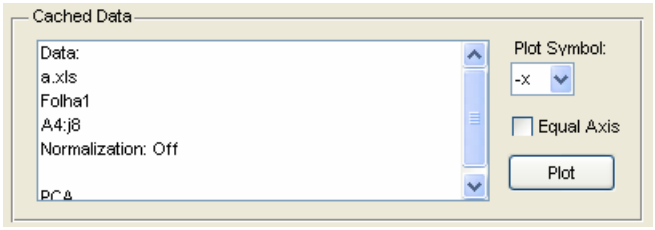
Principal Component Analysis is a statistic based method to reduce the number of dimensions of a data matrix. A window appears with points plotted on 3 graphs.

In the first scale of axes are adjusted, in the second graph 'equal axes' mode is on, and on the third graph there is a projection of the original variables, in order to allow a visualization of the contribution of each of the original variables for each new principal component.

(For further information, see *princomp* in Matlab Help)

Ploting Data

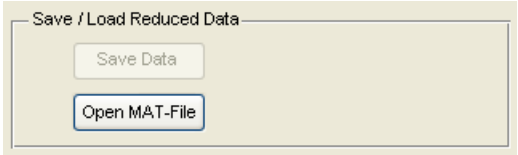
Cached Data panel shows information regarding the data that is currently on analysis.



By pressing the *Plot* button, it is possible to view the data represented by points in 2D or 3D (it depends on the amount of dimensions that were generated). The user may also choose the plot symbol, keeping in mind that symbols with minus signal (-) will have a line connecting the points sequentially.

Equal Axis option allows to disable auto sizing of figure axis.

Save and Load Reduced Matrixes



It is possible to save data processing (with respective data labels, if existing) to a file, along with the description present at the time in the *Cached Data* window. The user can choose the name and location of destination file in a window that pops up.

This file contains structure that holds all labels, the originally imported data and the reduced data (with eigen values if they exist).

Reading from a MAT file

One can import matrix that have already been reduced and saved with this interface, avoiding the necessity of recalculate the reduced matrix, which, depending on the amount of data and the methods used, can take from a few minutes to several days. This can be done just by pressing the *Open MAT-File* and selecting a *.mat* file. If the matfile was not saved by this program (does not contain some specified fields), an error message will appear.

Labels and information of the data and processes applied to the analysis were also stored during the save process, and are now read and available.

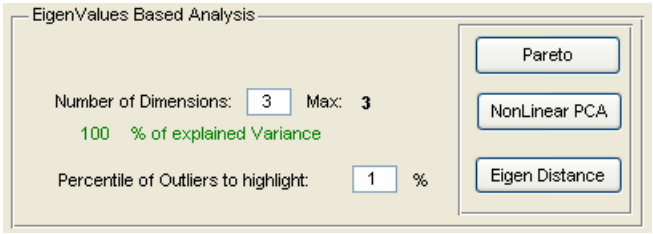
When a user opens a file that contains more than one object, the first matrix is loaded, and a list with the existing variables on the file appears.

Matrix with only 1 dimension or other types of unrecognized variables will not be loaded and a error message will appear.

EIGEN VALUES BASED ANALYSIS

Eigen Values Based Analysis

This panel allows several operations based on the existence of Eigen Values



Pareto button plots 2 graphics on a window: a bar graph for each of the first five eigen values, with a cumulative curve for them, and a curve for all eigen values. The datacursor mode (explained on the plot section) will show the principal component index and value.

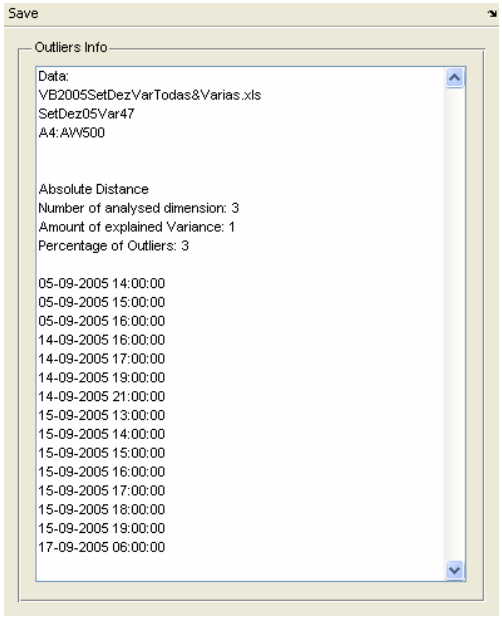
It is advisable to watch carefully for medium negative values of eigen values, since that is an indicator of poor quality of the performed reduction.

NonLinear PCA opens a second GUI that allows to calculate nonlinear principal components accordingly to the principle of bottleneck neural networks used for dimensions reduction. This will be explained in more detail in the next chapter.

EigenDistance calculates the absolute distance from points to origin, based on the principle that distances along each of the dimensions are proportional to the amount of variance that those dimensions represent.

For this calculation, '*Percentil of Outliers to highlight*', will set the percentage of existing further points to detect, and '*Number of Dimensions*' defines the number of dimensions that will be analysed.

The output will be presented in a window as depicted below, and can be saved to a file. The value of amount of explained variance is normalized between 0 and 1.

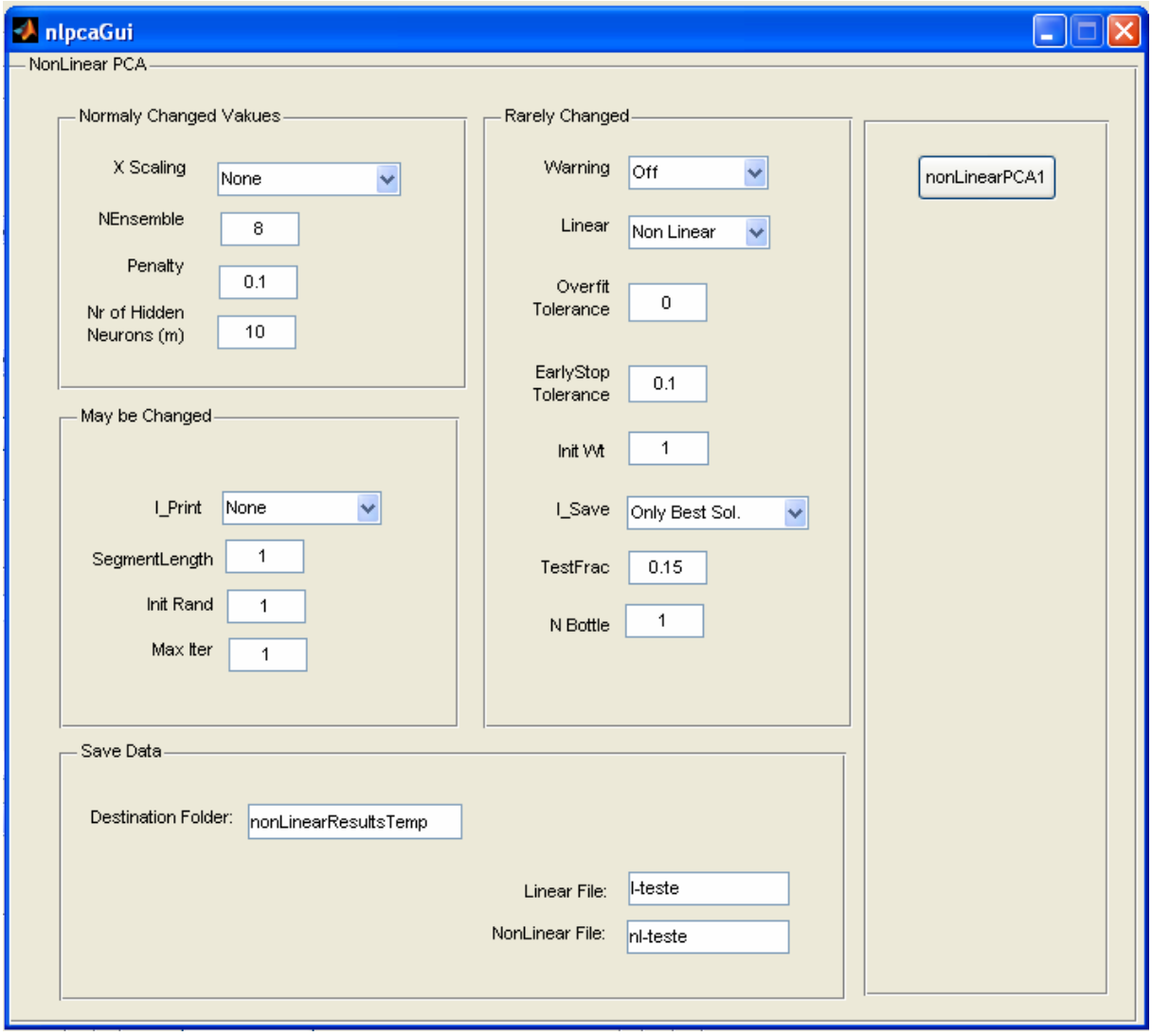


NON LINEAR PRINCIPAL COMPONENT ANALISYS

Non Linear PCA

Nonlinear principal component analysis is applied to the *n* first principal components, where *n* is a number or variables which sum of normalized eigen values is above 0.99 (this means that data explains at least 99 % of variance).

The developed interface implements the existing code of William W. Hsieh on his Neuralnets for Multivariate And Time Series Analysis (NeuMATSA) work. Below is depicted a figure of the developed GUI.



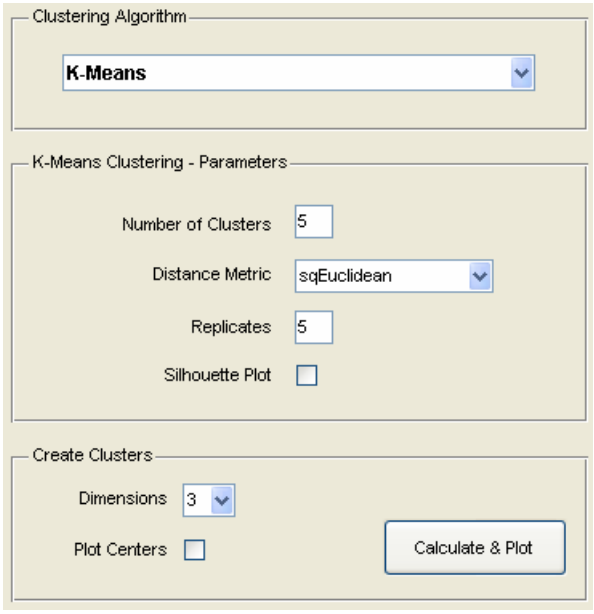
In it the user can easily manipulate several parameters that will affect de final bottle-neck neural network which is use to reduce dimensions. For a detailed description of which variable you should read the original user guide by William W. Hsieh.

Each of the resulting principal components is stored with a distinct name on the designated destination folder, which allows the data to be reviewed.

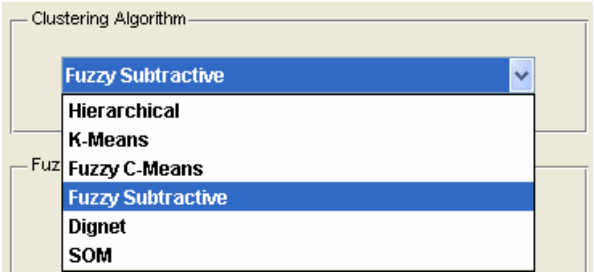
CLUSTERING

Clustering Algorithm

Clustering is enabled after computing dimension reduction (*Calculate* button on *Variable Reduction Method* panel), or when a reduced matrix is loaded from a recognizable *.mat* file. At this step the three panels that allow clustering control are enabled:



Clustering Algorithm panel allows choosing one of the six available clustering methods:

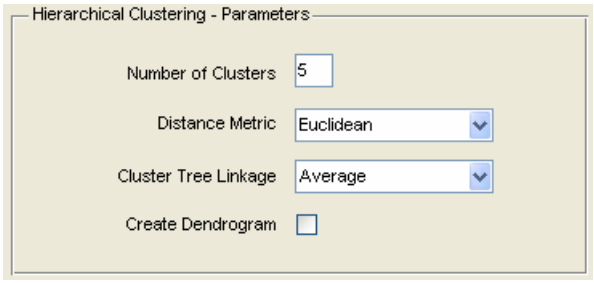


Depending on the choice of clustering algorithm, parameters panel presents related variables to current method, which are user controllable.

When reduced data is obtained by CMD or loaded from a file matrix with more than two dimensions, it is possible to set the number of dimensions to use in clustering process (in *Create Clusters* panel).

Next, a description of each method parameters is presented (by default, each method is set with values that produced the best clustering results).

1. Hierarchical Clustering



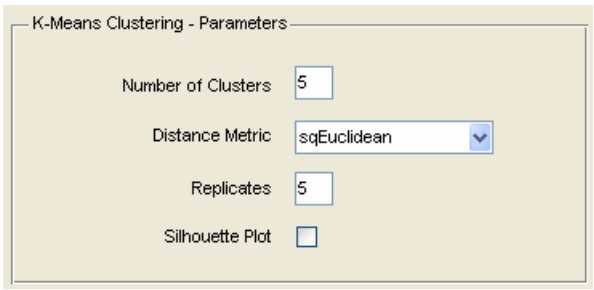
- *Number of Clusters*: number of groups to create.

- *Distance Metric*: metric used to compute distance between points. It can assume one of the six following values (for more information, refer to *pdist* function in Matlab Help):
  - Euclidean
  - Seuclidean
  - Mahalanobis
  - Cityblock
  - Chebychev
  - Minkowski (p=9)

- *Cluster Tree Linkage*: algorithm used to compute the hierarchical cluster tree. It can assume one of two values (for more information, refer to *linkage* function in Matlab Help):
  - Average
  - Complete

- *Create Dendrogram*: allows user to create or not a dendrogram plot. A dendrogram consists of many U-shaped lines connecting objects in a hierarchical tree. The height of each U represents the distance between the two objects being connected (for more information, refer to *dendrogram* function in Matlab Help).

2. K-Means Clustering



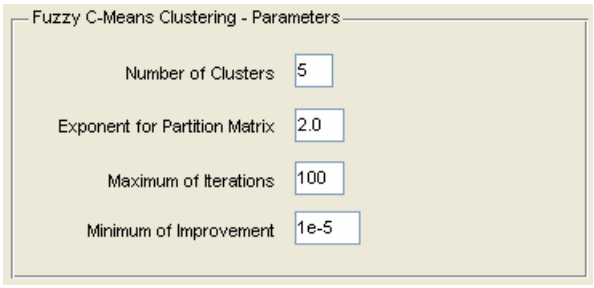
- *Number of Clusters*: number of groups to create.

- *Distance Metric*: defines distance measure to compute distance. This algorithm computes centroid clusters differently for the different supported distance measures. It can assume one of the three following values (for more information, refer to *kmeans* function in Matlab Help):
  - sqEuclidean
  - Cityblock
  - Cosine

o *Replicates*: number of times to repeat the clustering, each with a new set of initial cluster centroid positions. It is returned the solution with the lowest centroid distance sum (for more information, refer to *kmeans* function in Matlab Help).

o *Silhouette Plot*: allows user to create or not a silhouette plot. Silhouette plot focus the distance between neighbor clusters, setting lower values to points closer to neighbor clusters (for more information, refer to *silhouette* function in Matlab Help). Silhouette is plotted using the interpoint distance measure specified in *Distance Metric*.

3. Fuzzy C-Means Clustering



o *Number of Clusters*: number of groups to create.

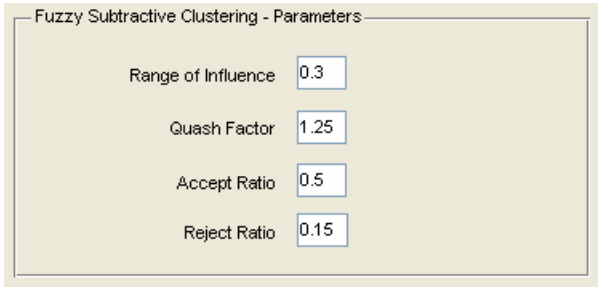
o *Exponent for Partition Matrix*: defines exponent for the partition matrix U.

o *Maximum of Iterations*: maximum number of iterations; computing is stopped when centroids distance sum doesn't decrease significantly between iterations.

o *Minimum of Improvement*: defines algorithm sensibility; clustering is stopped when the objective function improvement between two consecutive iterations is less than the minimum amount of improvement specified.

For more information on this parameters, refer to *fcm* function in Matlab Help.

4. Fuzzy Subtractive Clustering



o *Range of Influence*: variable between 0 and 1 that specifies cluster center's range of influence, assuming the data falls within a unit hyper box. Small values commonly result in finding a few large clusters. Good values are usually between 0.2 and 0.5.

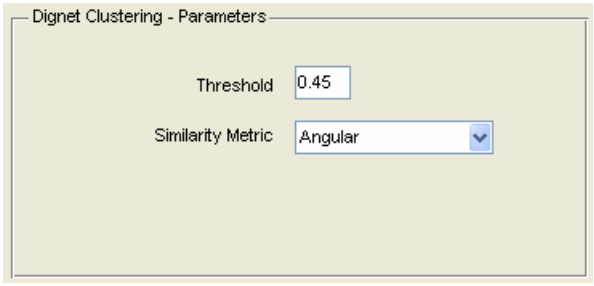
o *Quash Factor*: factor used to multiply the radii values that determine the neighborhood of a cluster center.

o *Accept Ratio*: potential, as a fraction of the potential of the first cluster center, above which another data point will be accepted as a cluster center.

o *Reject Ratio*: potential, as a fraction of the potential of the first cluster center, below which a data point will be rejected as a cluster center.

For more information on this parameters, refer to *subclust* function in Matlab Help.

5. Dignet Clustering



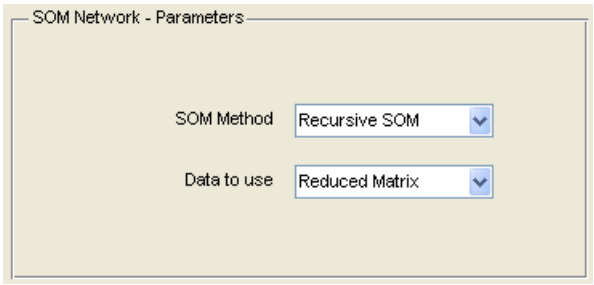
o *Threshold*: defines noise threshold used to compute clusters centers; small values result in finding a large number of clusters.

o *Similarity Metric*: defines distance measure to compute distance between cluster centers and data points. It can assume one of four values:

- Angular: $S_{j,n} = \arccos\left(\frac{\langle e_{j,n-1}; x_n \rangle}{\|e_{j,n-1}\| \cdot \|x_n\|}\right)$
- Euclidean: $S_{j,n} = \sqrt{\sum_k \left(e_{j_k,n-1} - x_{k,n}\right)^2}$
- Chebychev: $S_{j,n} = \max_k \left|e_{j_k,n-1} - x_{k,n}\right|$
- Cityblock: $S_{j,n} = \sum_k \left|e_{j_k,n-1} - x_{k,n}\right|$

On this method, $S_{j,n}$ defines similarity, e defines the cluster center that is being computed and x defines the point that is being measured.

6. SOM Clustering



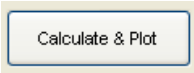
o *SOM Method*: defines SOM network method. Classical SOM and Recursive SOM are available.

o *Data to use*: allows user to choose which matrix to use (original or reduced data).

Tip: those clustering methods work on a different way comparing to classical clustering methods: this panel button opens a new interface window where it is possible to control and to visualize training and classification process.

Clustering method compute

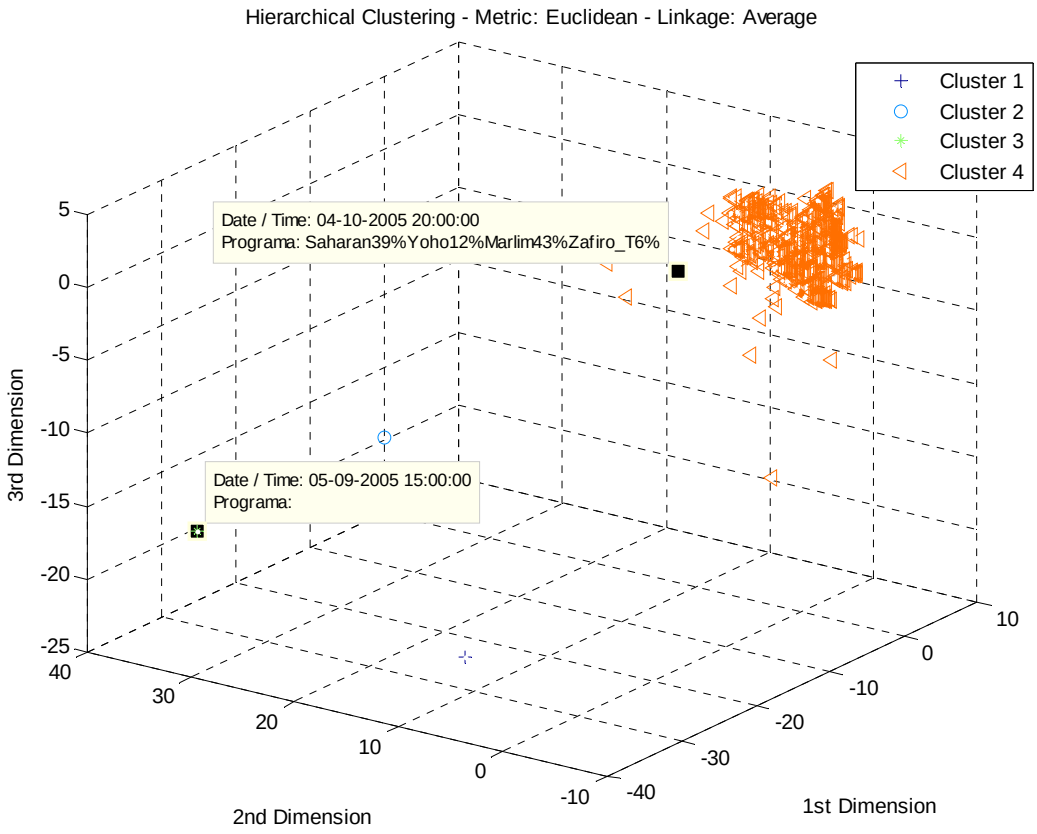
Description on this section is valid for all clustering methods, except SOM methods, once these are computed on specific windows. Clustering is computed by clicking *Calculate & Plot* button on *Create Clusters* panel:



So, clustering is done and the respective plot is generated (3D or 2D, as defined on this panel or on *MDS – Parameters* panel). In this graph each cluster is represented with distinct color and symbol.

Created plots have custom titles, with some important aspects of used clustering method, as well as cluster legend. Using 'Data Cursor' button (enabled by default), it is possible to show each point date and time (and industrial program, if available on read data); these two labels are only possible examples for this study case, once that application loads all label columns that are available on selected Excel file. Besides 'Zoom', another interesting tool on 3D plots is 'Rotate 3D' that allows rotating plot and analyzing it according to coordinate pairs.

Example of a Hierarchical Clustering generated plot:

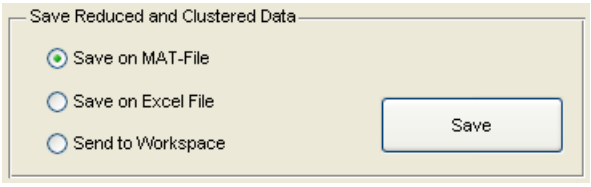


Plot title is *Hierarchical Clustering – Metric: Euclidean – Linkage: Average*, which shows applied clustering method and the most important defined parameters.

In this case there are two visible point labels (pushing Alt key allows showing several point labels at a time).

SAVING RESULTS

It is possible to save GUI generated information in *.mat* format, into an Excel file or in Matlab *Workspace*. Thus, it is only necessary to select destination and click *Save* button on *Save Reduced and Clustered Data* panel:



Save on MAT-File

According to what was described for *Save Data* button, it is also possible to store dimension reduction and clustering results into structures and save them on a *.mat* file.

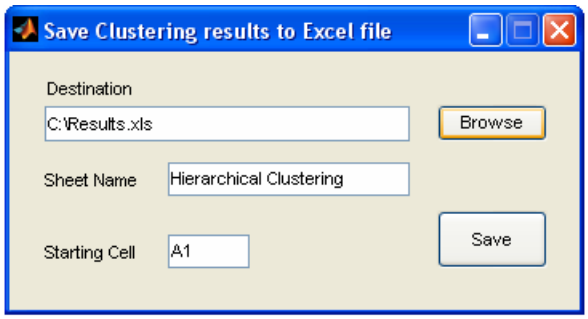
By clicking *Save* button with selected *Save on MAT-File* field, user can choose both name and localization of destination file, which will contain three or four structures (if data has been read from a file that was not created by the application, or from an Excel file):

1. *rawData*: structure containing Excel read data, including each normalization method result;
2. *labels*: structure containing data labels, including variable names;
3. *reducedData*: structure containing reduced data, including several important parameters;
4. *clusteringData*: structure containing clustering results, including parameters information.

Saved file can thus be loaded on *Save/Load Reduced Data* panel allowing reduced data analysis, or on *Load Clustered Data* panel allowing clusters visualization. In both cases no loss of important information occurs.

Save on Excel File

By clicking *Save* button with selected *Save on Excel File* field, the *Save Clustering results to Excel File* window is shown:



In this window it is possible to define destination file (using browser or typing file address in *Destination* box). Next, worksheet must be set (which name is the applied clustering method, by default) as well as the initial cell (left superior) where user wants to store data; if selected file or typed worksheet doesn't exist, they will be created by clicking *Save* button.

Data is stored according to the following configuration:

	A	B	C	D	E	F	G
1	Date / Time	Program	1st Dimension	2nd Dimension	3rd Dimension		Clusters
2	01-09-2005 0:00	Saharan39%Marlim54%EA7%	0,2913	-0,8247	0,8724		4
3	01-09-2005 1:00	Saharan39%Marlim54%EA7%	0,7137	-0,7183	1,2119		5
4	01-09-2005 2:00	Saharan39%Marlim54%EA7%	0,3503	-0,9914	0,8192		5
5	01-09-2005 3:00	Saharan39%Marlim54%EA7%	0,3427	-0,6286	0,3118		5
6	01-09-2005 4:00	Saharan39%Marlim54%EA7%	-0,7635	0,3428	-1,6859		2
7	01-09-2005 5:00	Saharan39%Marlim54%EA7%	0,2771	-1,2650	-1,5157		4
8	01-09-2005 6:00	Saharan39%Marlim54%EA7%	0,7055	-1,5091	-1,2432		4
9	01-09-2005 7:00	Saharan39%Marlim54%EA7%	0,4406	-1,1958	-1,5893		4
…	…	…	…	…	…		…

Column A contains first label column information (date and time by default) created on *Read* from *Read Raw Data* panel (or loaded from an application generated *.mat* file).

Column B contains, in this case, industrial information for each data entry; those labels are created from the second imported column on *Read* from *Read Raw Data* panel, if this column is of non-numeric type. If more label columns were read from Excel file, they are also stored.

Columns C, D and E contain reduced data dimensions (if dimension reduction was computed in 2D space, only columns C and D are filled).

Column G contains clustering index for each data entry.

Send to *Workspace*

By clicking *Save* button with selected *Send to Workspace* field, user stores data structures (already described at *Save on Mat-File* section) in Matlab *Workspace*.

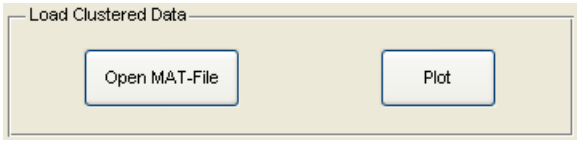
Then it is possible to immediately get all application data.

Tip: data to save includes cached and generated structures the last time *Calculate & Plot* button was clicked, after selecting clustering method and parameters.

Load clustered data

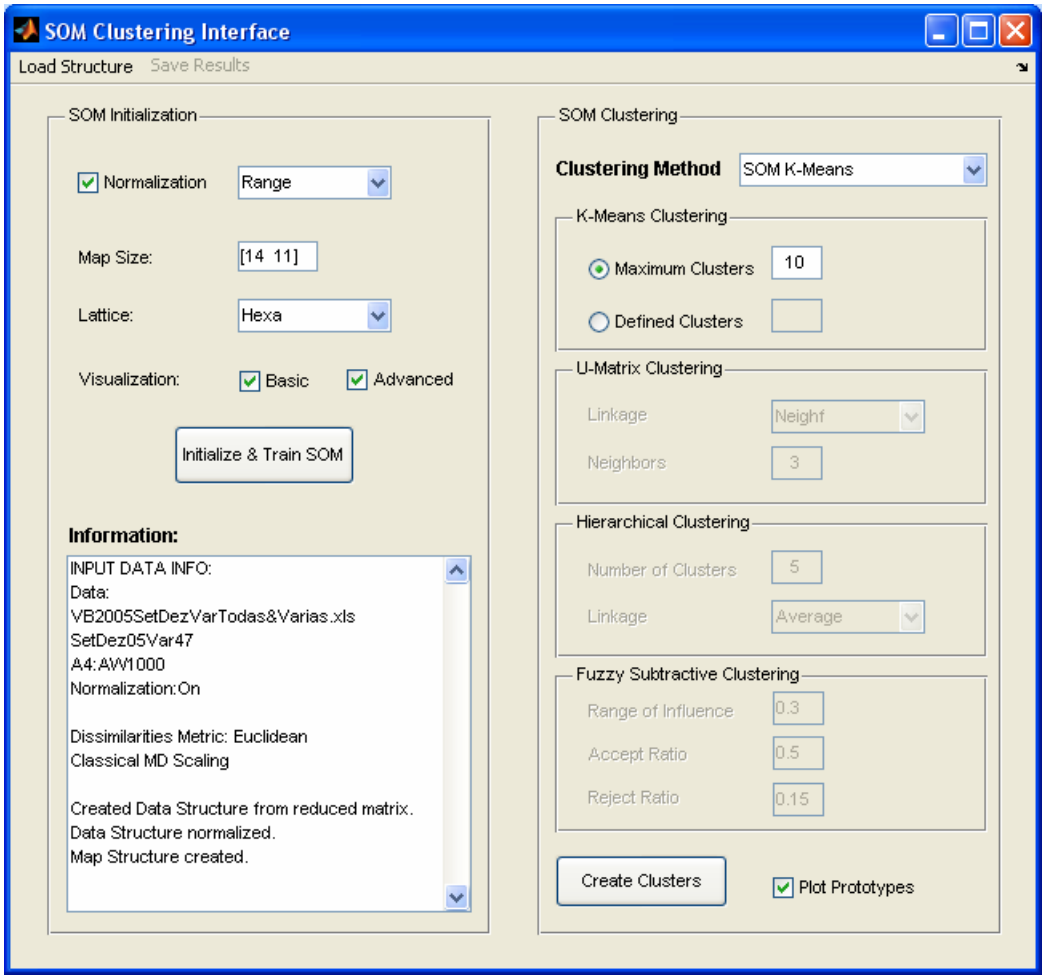
Open MAT-File button on *Load Clustered Data* panel allows loading *.mat* files containing clustering results that were saved by this application. If selected file doesn't contain necessary structures (*reducedData* and *clusteringData* structures) an error message is displayed: *MAT-File must contain reduced data and clustering structures*.

When a valid data file is loaded, *Plot* button of this panel is enabled, allowing data plotting with clusters and labels representation.



SOM CLUSTERING INTERFACE

SOM Clustering Interface window opens when user clicks *Open SOM Guide* button on *Create Clusters* panel, after selecting SOM Clustering method:



Self-Organizing Map (SOM), proposed by T. Kohonen, is an unsupervised neural network method which has properties of both vector quantization and vector projection algorithms. The prototype vectors are positioned on a regular low-dimensional grid in an ordered fashion, making the SOM a powerful visualization tool.

The SOM can be thought of as a net which is spread to the data cloud (as prototype vectors). The SOM training algorithm moves the weight vectors so that they span across the data cloud and so that the map is organized: neighboring neurons on the grid get similar weight vectors.

SOM Initialization

On *SOM Initialization* panel it is possible to control several SOM train parameters:

- *Normalization*: when selected, normalization is applied to given original or reduced data, using one of three methods: *Range* (values are normalized between 0 and 1), *Var* (variance is normalized to one), *Log* (natural logarithm is applied to the values: $x = \log(x - \min(x) + 1)$).

- *Map Size*: map grid size. This parameter is normalization dependent but it can be set by user.

- *Lattice*: topology, or structure, of the SOM. It can be defined by: *Hexa* (hexagonal shape) or *Rect* (rectangular shape).

- *Visualization*: it is possible to select two kinds of SOM visualization: *Basic* (shows the *U-Matrix*, a unified distance matrix of a SOM that focus the distance between map units) and *Advanced* (shows a four subplots window: a *U-Matrix* with map units sized according to the number of matching points, a surface distance matrix, a prototype plot and a prototype/data mixed plot).

- *Initialize & Train SOM* button: SOM train is performed, generating selected visualization methods and enabling clustering methods in *SOM Clustering* panel.

Tip: *Information* field displays important info on performed actions.

SOM Clustering

SOM clustering is applied to the prototype set computed in SOM train. Then, each original point is matched to the closest prototype cluster index.

Clustering Method allows four different clustering algorithms:

1. SOM K-Means

   - *Maximum Clusters*: maximum number of clusters. Algorithm selects the clustering number with smallest Davies-Bouldin index.

   - *Defined Clusters*: number of clusters to compute, regardless Davies-Bouldin index.

2. SOM U-Matrix

   - *Linkage*: defines distance method to compute distance between *U-Matrix* elements.

   - *Neighbors*: number of neighbors to extend distance measure.

3. Hierarchical Clustering

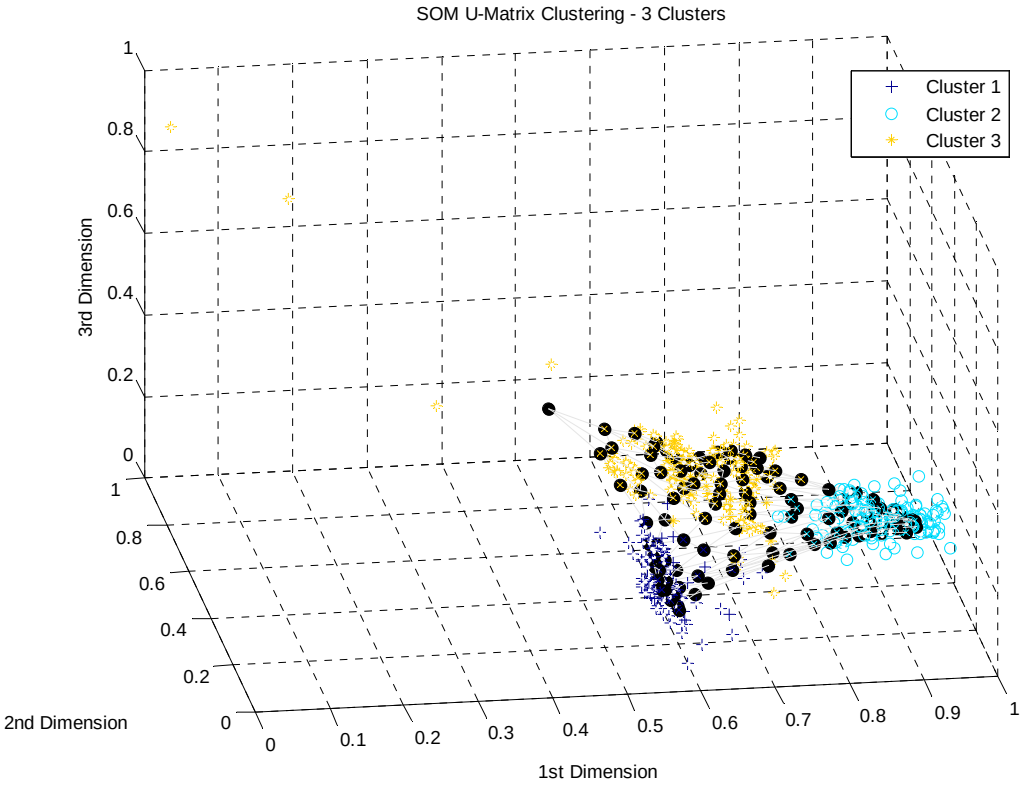
   Parameters described in *CLUSTERING* section.

4. Fuzzy Subtractive Clustering

   Parameters described in *CLUSTERING* section.

After choosing clustering method and respective parameters, user should click on *Create Clusters* button in order to proceed clustering and results plot.

Plot Prototypes selection box allows user to define if prototype vectors should be plotted. The final plot (including prototypes) looks like it follows:



Black linked points represent prototypes and color symbol points represent clustered original or reduced data. If original data has more than three dimensions, a 3D PCA projection of data is plotted.

Save SOM Results

User can save SOM clustering results on *.mat* and Excel files or send them to Matlab *Workspace*, by clicking on *Save Results* menu.

When *Save to MAT-File* submenu is clicked, user can choose name and path for the saving file, which will contain all the structures described on *Saving Results* section, including sD and sM structures, inside *clusteringData* structure. *Send to Workspace* method stores the same structures.

Clicking on *Save to Excel File* submenu, *Save Clustering results to Excel File* window is shown. This window was already described on *Saving Results* section.

Load SOM Structure

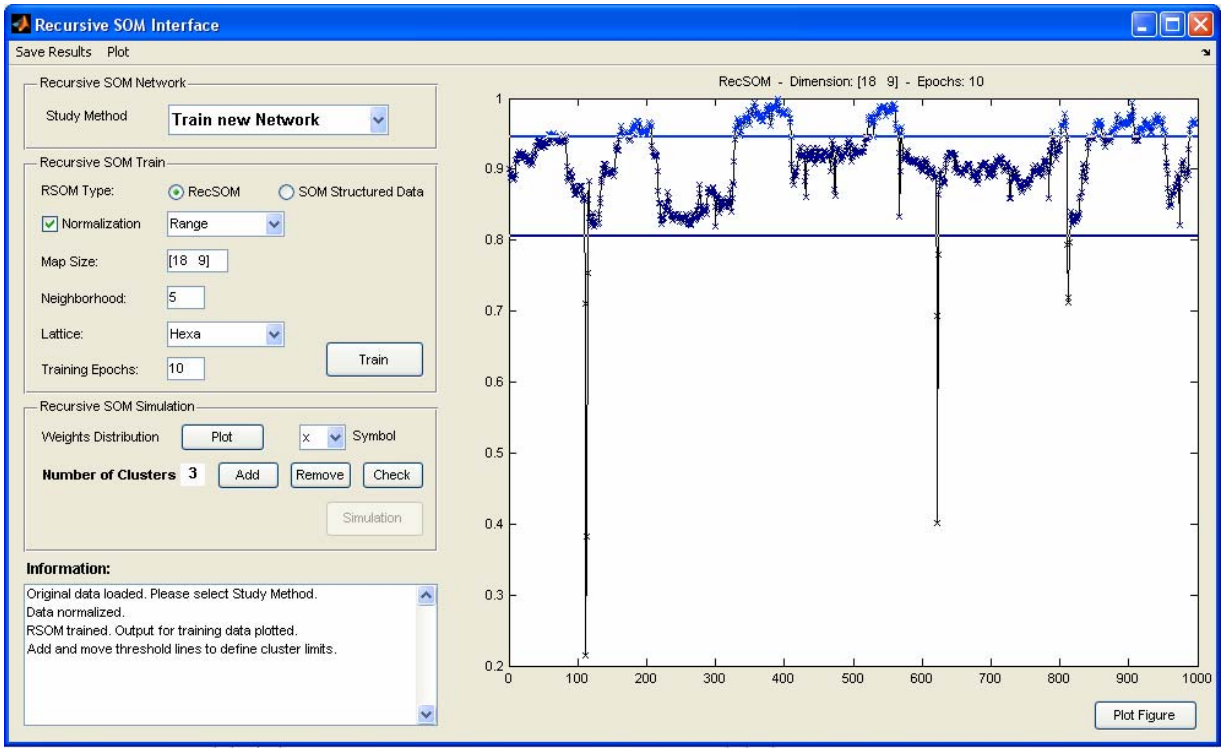
User can load *.mat* files containing SOM structures (*data structure* – sD – and *map structure* – sM) by clicking on *Load Structure* menu.

If an sM structure is loaded, *SOM Initialization* panel is disabled and SOM clustering methods become available.

If an sD structure is loaded, *SOM Clustering* panel is disabled and SOM initialization training methods become available.

RECURSIVE SOM INTERFACE

Recursive SOM Interface window opens when user clicks *Open RecSOM Guide* button on *Create Clusters* panel, after selecting Recursive SOM clustering method:



The Recursive SOM network is a generalization of SOM that learns to represent sequences recursively. Its resulting representations are adapted to the temporal statistics of the input series.

The SOM Structured Data network is also an extension of the standard SOM model, which allows the mapping of structured objects into a topological map. This mapping can then be used to discover similarities among the input objects.

Implementation of this interface is based on Recurrent Self-Organizing Map (RSOM) Toolbox for MATLAB, developed by Amir Reza Saffari Azar Alamdari, July 2005.

Study method

Recursive SOM Network panel allows user to choose the intended study method to compute given interface data:

- *Train new Network*: allows user to train a SOM network using current data (original or reduced data);

- *Load trained Network*: a file selection window is shown, where one can load a network trained by this interface; loaded net can then be simulated using given interface data as its input. Net output for training data and cluster partition are plotted at interface plot area.

Recursive SOM Train

Recursive SOM Train panel, only available on *Train new Network* study method, allows training parameters definition. Some of these parameters were already described on *SOM CLUSTERING INTERFACE* section, so following ones can be distinguished:

- *RSOM Type*: allows user to choose network type (Recursive SOM or SOM Structured Data, described above);

- *Training Epochs*: defines the number of network training iterations; regard that Recursive SOM is much slower than SOM Structured Data

When *Train* button is clicked, network train is performed and plot area shows the one-dimensional output for training data.

Tip: *Information* field displays important info on performed actions.

Recursive SOM Simulation

Recursive SOM Simulation panel, only available after SOM training or loading, allows three main purposes:

- *Weights Distribution*: displays spatial sorting of trained (or loaded) network;

- *Number of Clusters*: allows setting cluster partition to apply to network output. One can add or remove clusters (between 1 and 8) and easily move colored cluster lines. *Check* button updates current cluster partition state by reformatting plot area. *Plot Figure* button plots the same results on a separated window;

- *Simulation* button: only available on *Load trained Network* method, computes net simulation for interface given data. Cluster partition is kept.

Tip: *Plot* menu contains plot analysis tools that are similar to Matlab plot window tools.

Save Recursive SOM Results

User can save SOM train and simulation results on *.mat* and Excel files or send them to Matlab *Workspace*, by clicking on *Save Results* menu.

When *Save to MAT-File* submenu is clicked, user can choose name and path for the saving file, which will contain all the structures described on *Saving Results* section, including network and cluster partition variables, inside *clusteringData* structure. *Send to Workspace* method stores the same structures.

Clicking on *Save to Excel File* submenu, *Save Clustering results to Excel File* window is shown. This window was already described on *Saving Results* section.