CLASSE – Synthetic Classification for Industrial Supervision

VISRED

Data Visualization by Space Reduction

Version 2.0

User Guide

António Dourado
Edgar Ferreira
Paulo Barbeiro

FCTUC

July 13, 2007

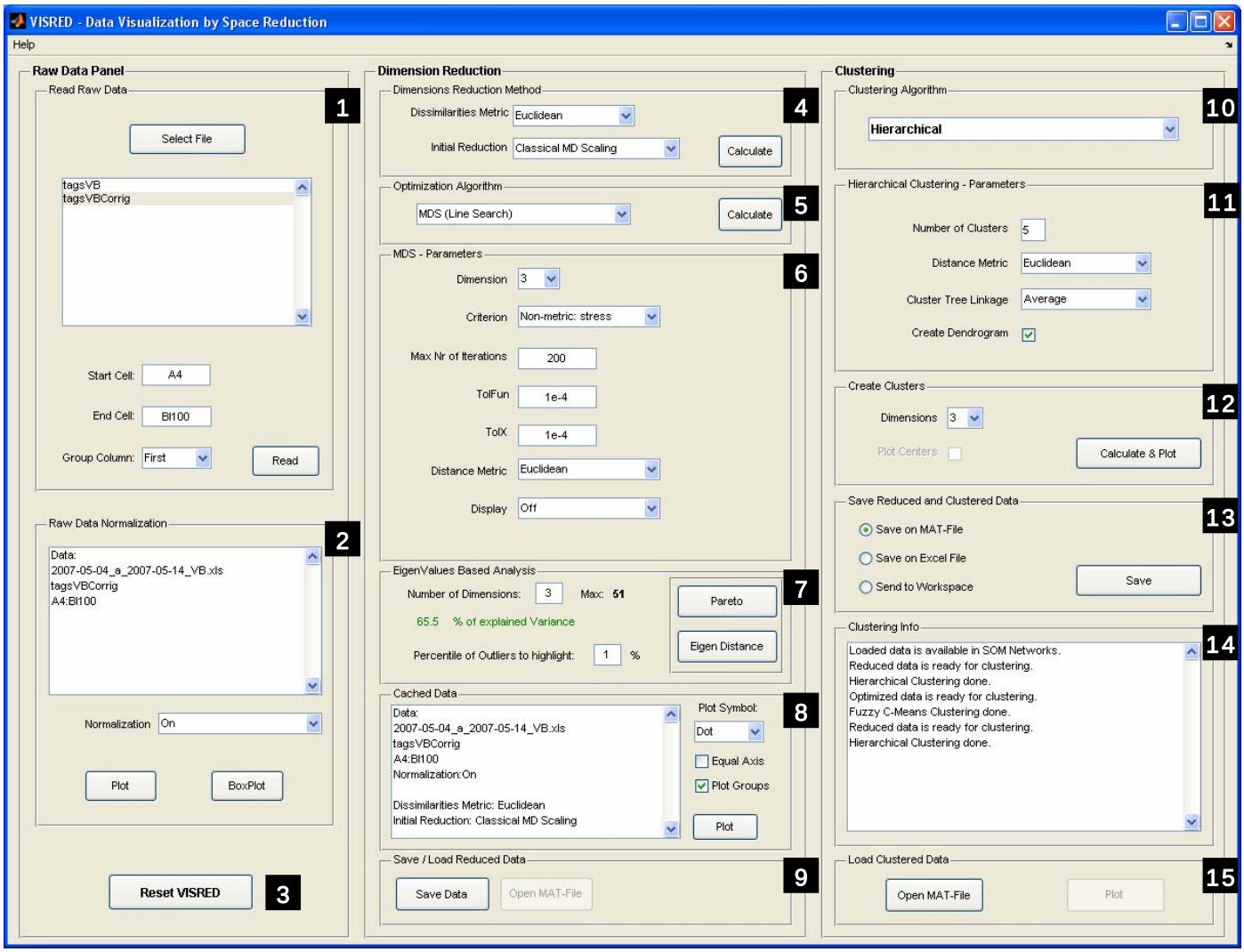
TABLE OF CONTENTS

Page

Authors:
Edgar da Silva Ferreira – edgar.bajouca@gmail.com
Paulo Filipe Domingues Barbeiro – paulobarbeiro@gmail.com

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. *Dimensions Reduction Method*: Methods to calculate dissimilarities and compute initial dimension reduction
5. *Optimization Algorithm*: Allows choosing the optimization method to use
6. *MDS - Parameters / Genetic Algorithm – Parameters / Simulated Annealing – Parameters*: Allow setting optimization parameters
7. *EigenValues Based Analysis*: Several methods of visualization based on existing Eigen values
8. *Cached Data*: Panel with information about data currently being analysed

9. *Save /Load Reduced Data*: Loads a file containing a reduced matrix, saved by this application
10. *Clustering Algorithm*: Allows choosing the clustering method to compute
11. *Clustering – Parameters*: Allows defining parameters of current clustering method
12. *Create Clusters*: Allows user to compute clusters
13. *Save Reduced and Clustered Data*: Allows user to save results to an Excel or MAT file, or to Matlab *Workspace*
14. *Clustering Info*: Displays clustering related information
15. *Load Clustered Data*: Contains buttons that allow user to load and plot clustered data

Note: plots are made on separated windows.

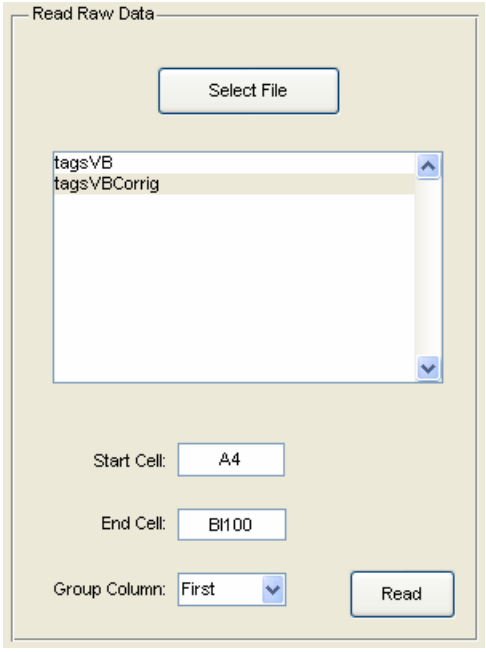
DATA IMPORT

Excel File Reading

Data that 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 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 *start cell* (left top corner of data) corresponds to the variable names line (if it exists, else it 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.

Group Column is a numeric column defined by user, which will allow plotting data in colored groups. Rows with the same number belong to same group. This column must appear after all labels, before or after the rest of data (ie, it must be the first or last column of 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 rows for *Headers* and *Variables* labels, they will all be concatenated to just one row.

Note**:** Last column of labels must contain text, but the other columns of labels don't have this restriction, as the algorithm is able to convert them to text.

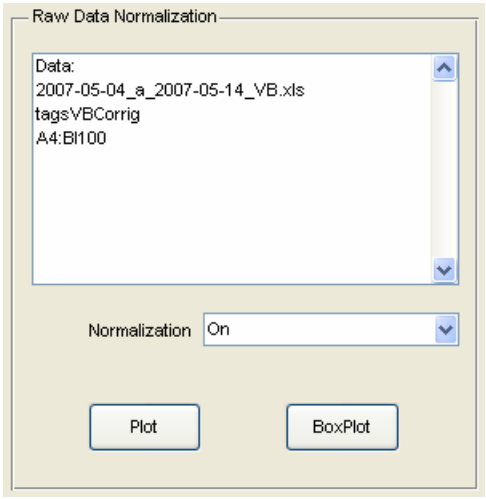
If data is incomplete (numeric cells without values), a warning message will appear and loading process will be halted.

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 may induce errors on the analyzing process. To prevent this from happening, normalization is required, although some important information may be lost during this process.

Method for 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

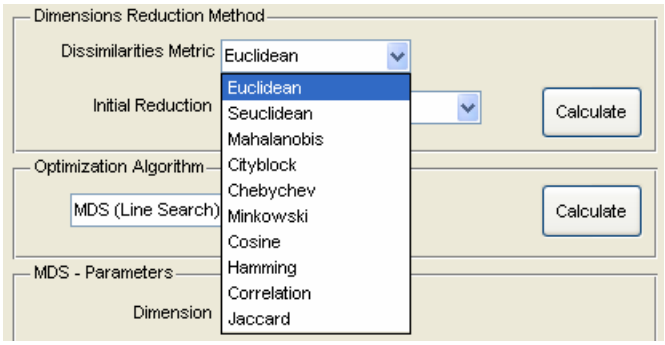
Plot button – Plots read data, with events on X axis and Variable Values on Y axis. Clicking on points of lines will show to which variable and event is selected.

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.

DIMENSIONS 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.



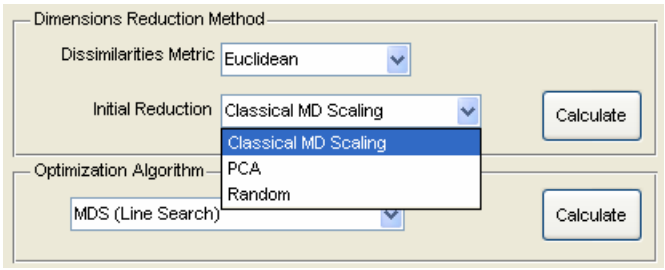
Note: From our experience, best results are obtained with Euclidean, SEuclidean, CityBlock and Correlation metrics.

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

Dimensions Reduction Method

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 the initial dimensions reduction can be chosen 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 an Eigen values vector, where each Eigen value corresponds to the representativeness of the each new dimension.

  (For further information, see *cmdscale* 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)

- Random

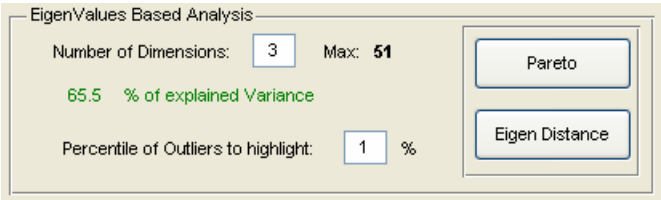
  This method creates a random matrix based on raw data statistics.

- Non Linear Principal Component Analysis (nlPCA)

  This method calls a Graphic User Interface (GUI) which allows adjusting parameters and calculating the several components of the nlPCA. Since this is also an optimization algorithm, variables are operation flow are explained on that section.

Eigen Values Based Analysis

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



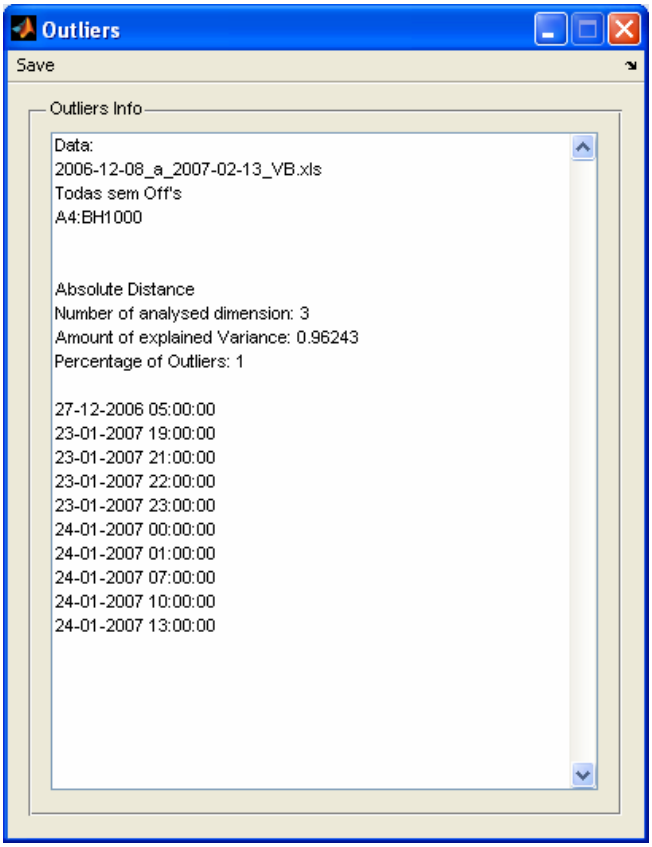
Pareto button plots two graphics on a window: a bar graph for each of the first five Eigen values, with a cumulative curve for them, and a curve for all Eigen values. The *datacursor* mode (explained on the plot section) will show the principal component index and value.

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.

Eigen Distance 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.

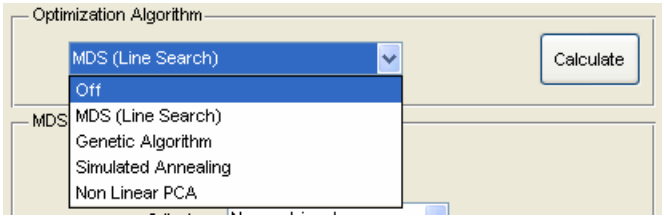
The output will be presented in a window as depicted below, and can be saved to a file. The value of amount of explained variance is normalized between 0 and 1.


Outliers
Save
Outliers Info
Data:
2006-12-08_a_2007-02-13_VB.xls
Todas sem Off's
A4:BH1000
Absolute Distance
Number of analysed dimension: 3
Amount of explained Variance: 0.96243
Percentage of Outliers: 1
27-12-2006 05:00:00
23-01-2007 19:00:00
23-01-2007 21:00:00
23-01-2007 22:00:00
23-01-2007 23:00:00
24-01-2007 00:00:00
24-01-2007 01:00:00
24-01-2007 07:00:00
24-01-2007 10:00:00
24-01-2007 13:00:00

OPTIMIZATION ALGORITHMS

Several optimization techniques can be used so as to improve Dimensions Reduction results. These methods are available on *Optimization Algorithm* panel and use data from initial reduction as starting reference, in order to improve defined criterion.

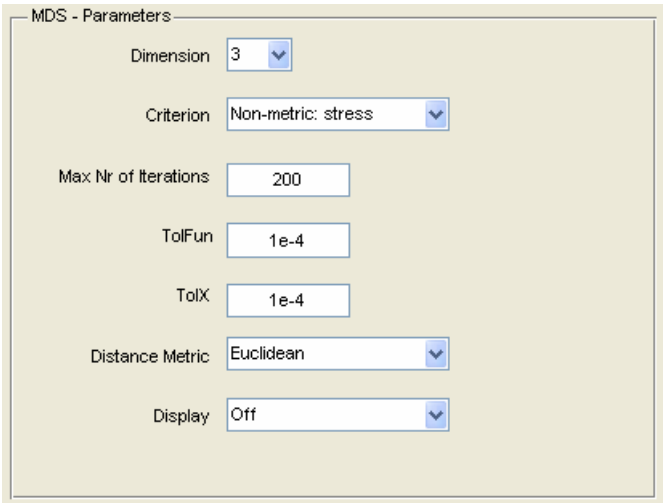
In every optimization algorithm, the ideal configuration of points would be the one whose distance between points is the same as a reference one. This reference configuration is calculated when the dimensions reduction is made, with the metric and normalization selected at the time.



Optimization methods can be performed sequentially, allowing many criterion improvements to a data sample.

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 a short description of them is made below.

Parameters:

Dimension: Desired number of dimensions for data after optimization process.

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

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.

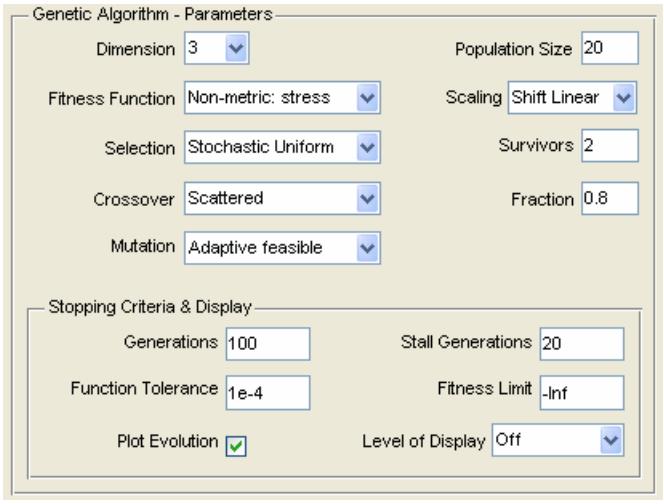
Display: Allows defining the amount of displayed output on Matlab's console.

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

Genetic Algorithm

The genetic algorithm is a method for solving optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population develops toward an optimal solution. [1]

Genetic algorithm panel is enabled by selecting *Genetic Algorithm* option on *Optimization Algorithm* panel:



This method is based on *ga* function from Genetic Algorithm and Direct Search Toolbox. Tens of variables can be modified using this function; however, only the most significant parameters (those that produce the highest behavior changes) can be changed on *Genetic Algorithm – Parameters* panel, as described below:

- *Dimension*: defines the number of dimensions for data after optimization process.

- *Population Size*: defines how many individuals there are in each generation.

- *Fitness Function*: defines the goodness-of-fit criterion algorithm will try to minimize on each generation.

- *Scaling*: converts raw fitness scores returned by the fitness function to values in a range that is suitable for the selection function.

[1] *Genetic Algorithm and Direct Search Toolbox User's Guide*, The MathWorks, Inc. 2007

o *Selection*: chooses parents for the next generation based on their scaled values from the fitness scaling method.
o *Survivors*: specifies the number of individuals that are guaranteed to survive to the next generation.

o *Crossover*: specifies how to perform the combination of two individuals, or parents, to form a new individual, or child, for the next generation.

o *Fraction*: specifies the fraction of the next generation that is produced by crossover. The remaining offspring in the next generation is produced by mutation.

o *Mutation*: specifies how to make small random changes in the individuals in the population, which provide genetic diversity.

o *Generations*: specifies the maximum number of iterations the genetic algorithm performs.

o *Stall Generations*: specifies the maximum computing generations without significant fitness improvement.

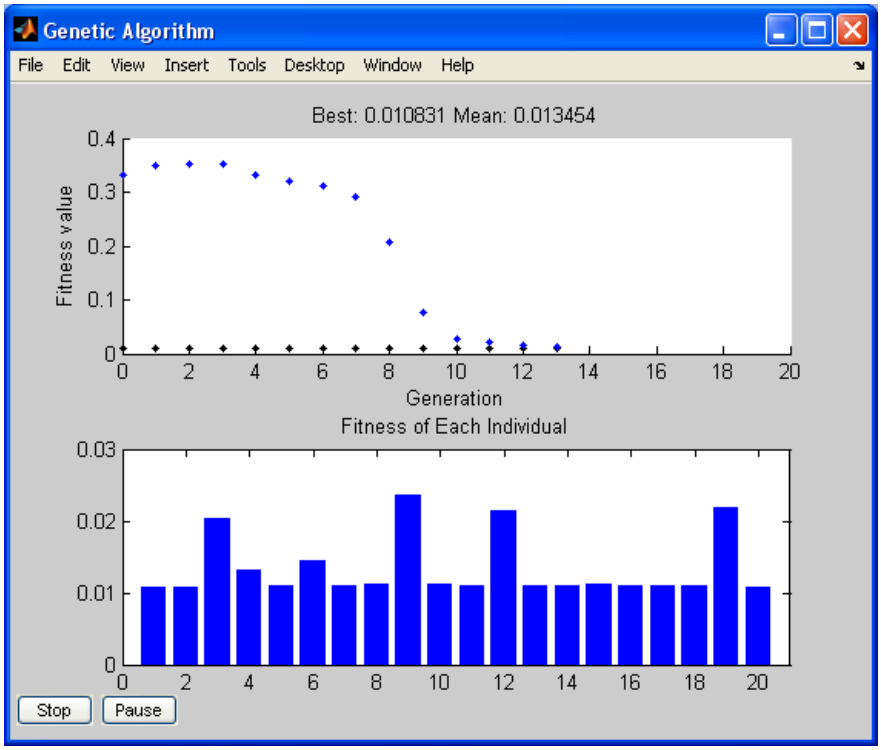
o *Function Tolerance*: specifies the minimum cumulative change in the fitness function value over stall generations.

o *Fitness Limit*: specifies the minimum achievable fitness value.

o *Plot evolution*: allows user to plot genetic algorithm evolution as it is executing. If selected, user can stop the process by clicking *Stop* button.

o *Level of Display*: specifies the amount of information displayed in the MATLAB Command Window.

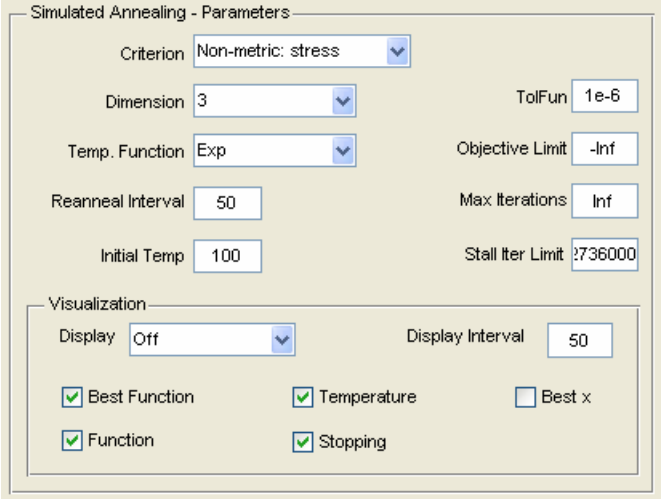
During genetic algorithm performing, if *Plot Evolution* is selected, an evolutionary plot is shown (overall evolution on top and iterative behavior on bottom):



When optimization is completed, a dialog box shows the improvement achieved by this algorithm and results are saved on *reducedData* structure.

Simulated Annealing

The name and inspiration of this technique come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one.[2]



This method is based on *simulannealbnd* function from Genetic Algorithm and Direct Search Toolbox. From all the variables that can be modified, the ones thought as more important were selected and can be changed on the *Simulated Annealing – Parameters*, as listed below:

- *Dimension*: defines the number of dimensions for data after optimization process.
- *Criterion*: defines the goodness-of-fit criterion algorithm will try to minimize on each generation.
- *Temperature Function*: Function used to update the temperature schedule
- *Initial Temperature*: Initial temperature at the start of the algorithm.
- *Reanneal Interval*: Number of points accepted before reannealing (increasing temperature)
- *Function Tolerance (TolFun)*: specifies the minimum cumulative change in the fitness function value over stall generations.
- *Objective Limit*: Minimum objective function value desired

[2] http://en.wikipedia.org/wiki/Simulated_annealing

- *Maximum Iterations*: Maximum number of iterations to execute before stopping.

- *Stall Iteration Limit*: Number of iterations over which average change in objective function value at current point is less than *Function Tolerance*
- *Display*: specifies the amount of information displayed in the MATLAB Command Window.

- *Display Interval*: specifies the interval for iterative display

- *Plot Functions*

    - *Best Function*: Best function value obtained

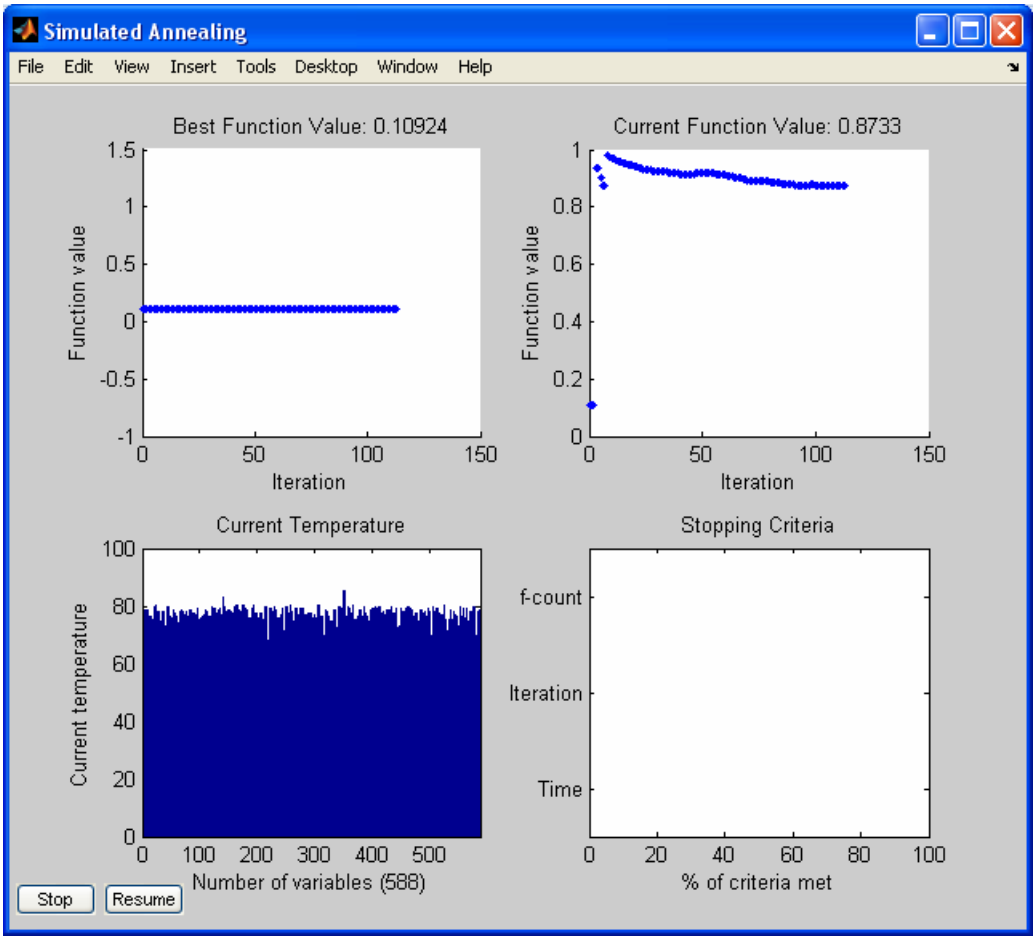
    - *Temperature*: Current temperature for each variable

    - *Best X* : Best point found so far (best X)

    - *Function*: Current function value

    - *Stopping*: Percentage of completed stopping criteria

For the default selected plot functions, the graphic that will appear is depicted below:



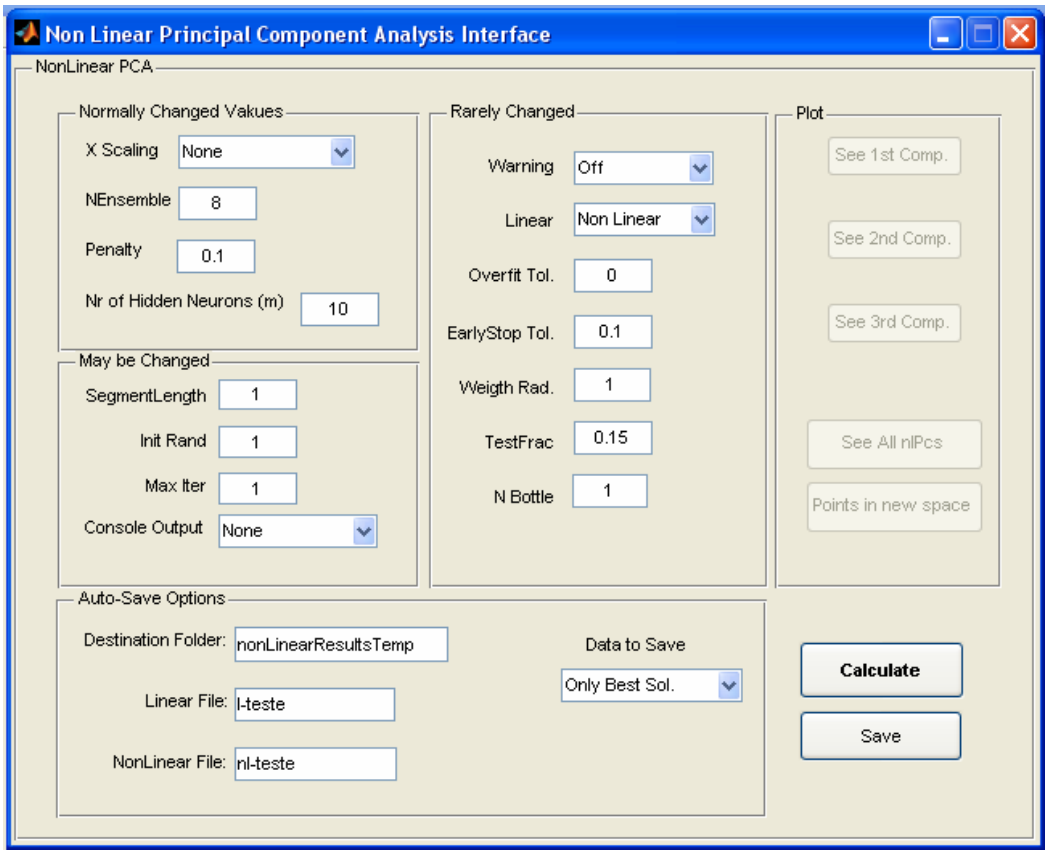
When optimization is completed, a dialog box shows the improvement achieved by this algorithm and results are saved on *reducedData* structure.

Non Linear PCA

Non Linear PCA opens a second GUI that allows calculating nonlinear principal components accordingly to the principle of bottleneck neural networks used for dimensions reduction.

Nonlinear principal component analysis optimization is applied to the *n* first principal components, where *n* is a number or variables which sum of normalized Eigen values is above 0.995 (this means that data explains at least 99.5 % 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 of the bottle-neck neural network which is used to reduce dimensions.

- *X Scaling* – Defines the scaling of each variable, wich can be none, or removal the mean and division by standard deviation.
- *NEnsemble* – Number of retraining of the NN. Since the sample to train and test is acquired randomly, increasing this value will increment the possibility of find a global minimum.
- *Penalty* – Scales the weight penalty term in the cost function. Increasing penalty leads to less nonlinearity.
- *Nr of Hidden Neurons* – Number of neurons in the hidden layer. Increasing this number will increase the nonlinearity of the possible solution

- *Segment Length* – Data for training and testing are chosen in segments with length equal to segmentlength (increase this value if correlation of data is a problem).
- *Init Rand* – Initializes random numbers generator.
- *Max Iterations* – Scales the maximum number of iterations allowed
- *Console Output* – Amount of information that will be displayed in the console.
- *Warning* – Allows user to choose if internal warnings will be shown or not.
- *Linear* – Define if the calculated PCA will be Linear or NonLinear.
- *Overfit Tolerance* – Tolerance for overfitting. When this value is zero, program automatically chooses an overfit tolerence level.
- *Early Stop Tolerance* – Should be around 0.05 to 0.15, and avoids overfit.
- *Weight Radius* – Controls the initial random weight radius for the ensemble member, and must be adjusted if there are convergence problems.
- *Test Fraction* – Fraction of data that will be reserved to NN testing.
- *N bottle* – Number of neurons in the bottle-neck layer. Although it is possible to use more than one neuron for calculation, visualization of this data must be hand-made by the user, since the interface only covers the possibility of using 1 neuron in the bottleneck layer (in order to obtain orthogonal components).

For a even more detailed description of each variable you should read the original user guide by William W. Hsieh.

Each time the *Calculate* button is pressed, a new NLPC is calculated, up to a maximum of 3 of them. Components can be views regarding the data used to retrieve them – *See n^{th} Comp.* –, the initial data that's being analyzed – *See all nlPcs* –, and the mapping of the original data to the new space.

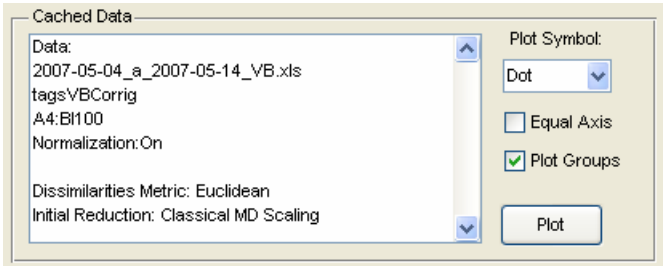
Due to the existing implementation, each of the resulting principal components is automatically stored with a distinct name on the designated destination folder, which allows the data to be reviewed. As these files are also needed to plot the components in the interface, so it's not advisable to change them while running the interface.

After finishing all calculations, pressing the save button will close the interface and send the points in the new space back to VisRed main interface. Closing the interface by other means will simply return to VisRed.

DATA VISUALIZATION AND MANAGEMENT

Plotting Data

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

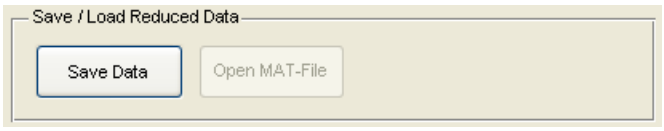


By pressing the *Plot* button, it is possible to view data represented by points in 2D or 3D (it depends on the amount of dimensions that were generated). User may also choose the plot symbol, keeping in mind that symbols with minus signal (-) will have a line connecting the points sequentially.

Equal Axis option allows disabling auto sizing of figure axis.

Plot Groups option is enabled when categorized data are loaded (as referred in *Data Import* chapter). When this option is checked, user can plot each group on a different color.

Save and Load Reduced Matrices

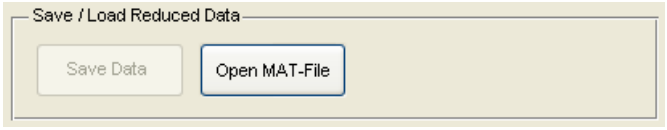


It is possible to save processed data (with respective data labels, if existing) into a file, along with the description present at the time in the *Cached Data* window. User can choose the name and location of destination file in a window that pops up.

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

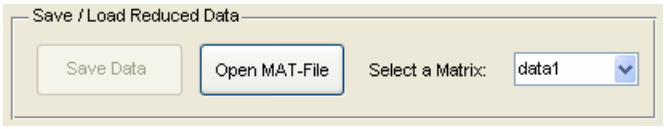
Reading from a MAT file

One can import matrices 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 used methods, can take from a few minutes to several days. This can be done by just pressing *Open MAT-File* button and selecting a *.mat* file. If the *.mat* file was not saved by this application (does not contain some specified fields), an error message will appear.



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

When user opens a file that contains more than one object, the first matrix is loaded and a list with the existing variables in the file appears (which means that this file was not created by this application).

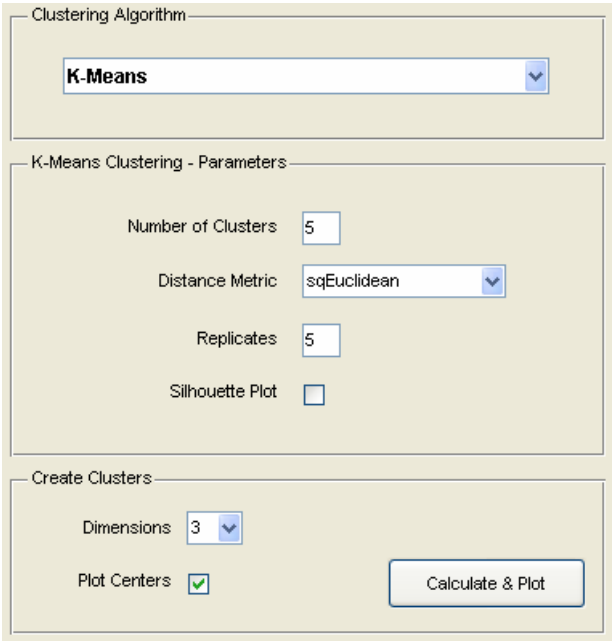


Matrices with only one dimension or other types of unrecognized variables will not be loaded and an error message will appear.

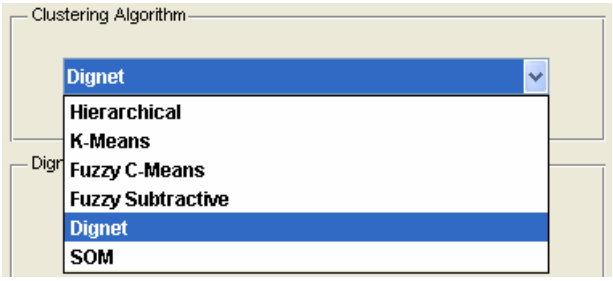
CLUSTERING

Clustering Algorithm

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



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

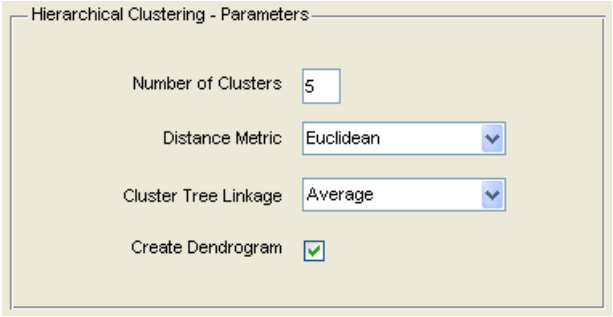


Depending on the choice of clustering algorithm, parameters panel presents variables related to current method, which are user controllable.

When reduced data is obtained by a reduction method 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



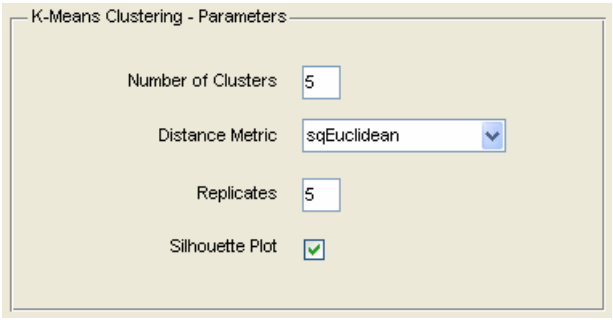
- *Number of Clusters*: number of groups to create.

- *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 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



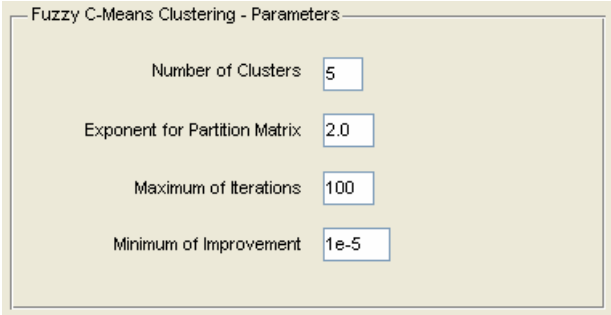
- *Number of Clusters*: number of groups to create.

- *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 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.

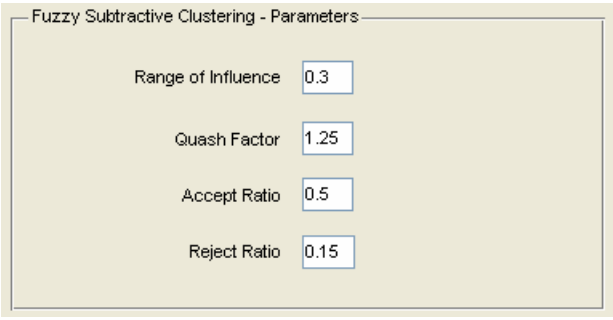
o *Exponent for Partition Matrix*: defines exponent for the partition matrix U.

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.

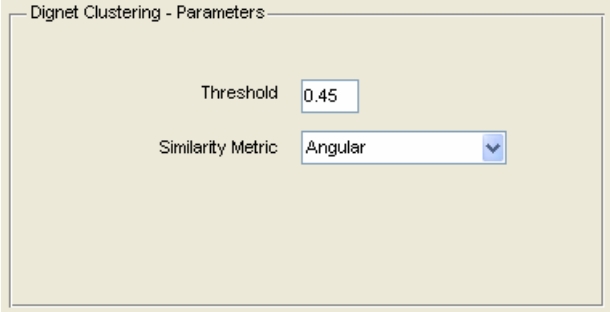
o *Quash Factor*: factor used to multiply the radii values that determine the neighborhood of a cluster center.

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



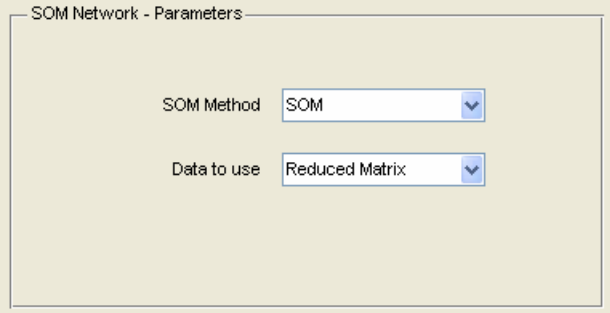
o *Threshold*: defines noise threshold used to compute clusters centers; small values result in finding a large number of clusters.

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: $S_{j,n} = \sum_k \left|e_{j_k,n-1} - x_{k,n}\right|$

In 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



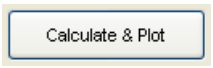
o *SOM Method*: defines SOM network method. Classical SOM and Recursive SOM are available.

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

Note: these clustering methods work on a different way when compared to classical clustering methods: a new interface window where it is possible to control and to visualize training and classification process is opened.

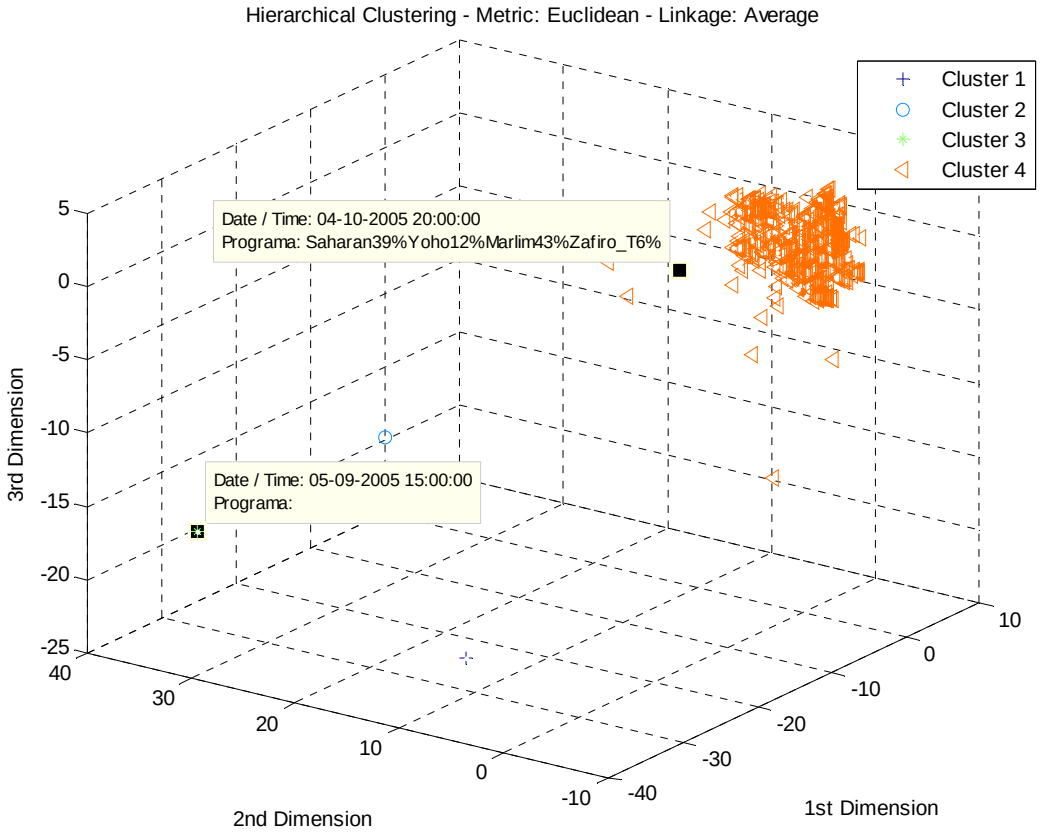
Clustering Method Calculation

Description in 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:



Thus, 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 a distinct color and symbol.

Created plots have custom titles, illustrating some important aspects of applied 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 because 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.



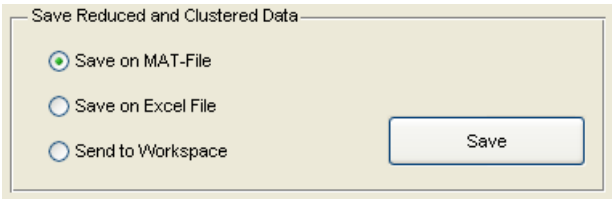
Example of a Hierarchical Clustering generated plot

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 the desired 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.

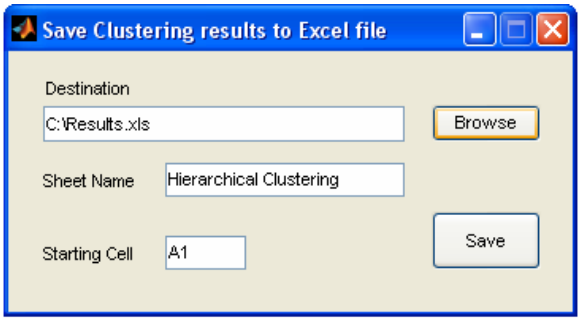
By clicking *Save* button when *Save on MAT-File* field is selected, user can choose both name and localization of destination file, which will contain three or four structures (if data has been read from a file that was not created by the application, or from an Excel file):

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 when *Save on Excel File* field is selected, 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, the 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 are 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 a *.mat* file created by this application).

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 are 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 when *Send to Workspace* field is selected, 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.

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

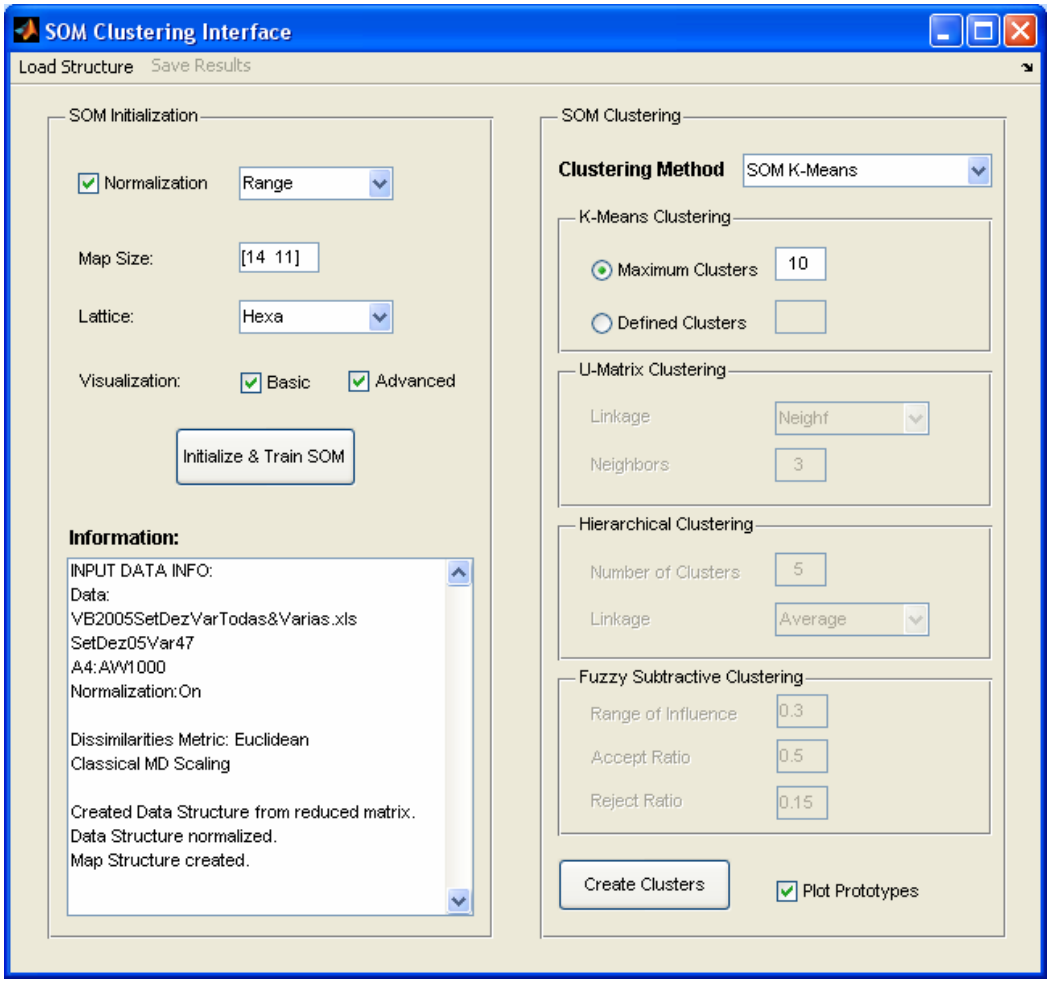
Load Clustered Data

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[3].

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: $x = \log(x - \min(x) + 1)$).

[3] Vesanto, J., Himberg. J, Alhoniemi, E. and Parhankangas, J. (2000), SOM Toolbox for Matlab 5, Helsinki University of Technology

- *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.

Note: *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

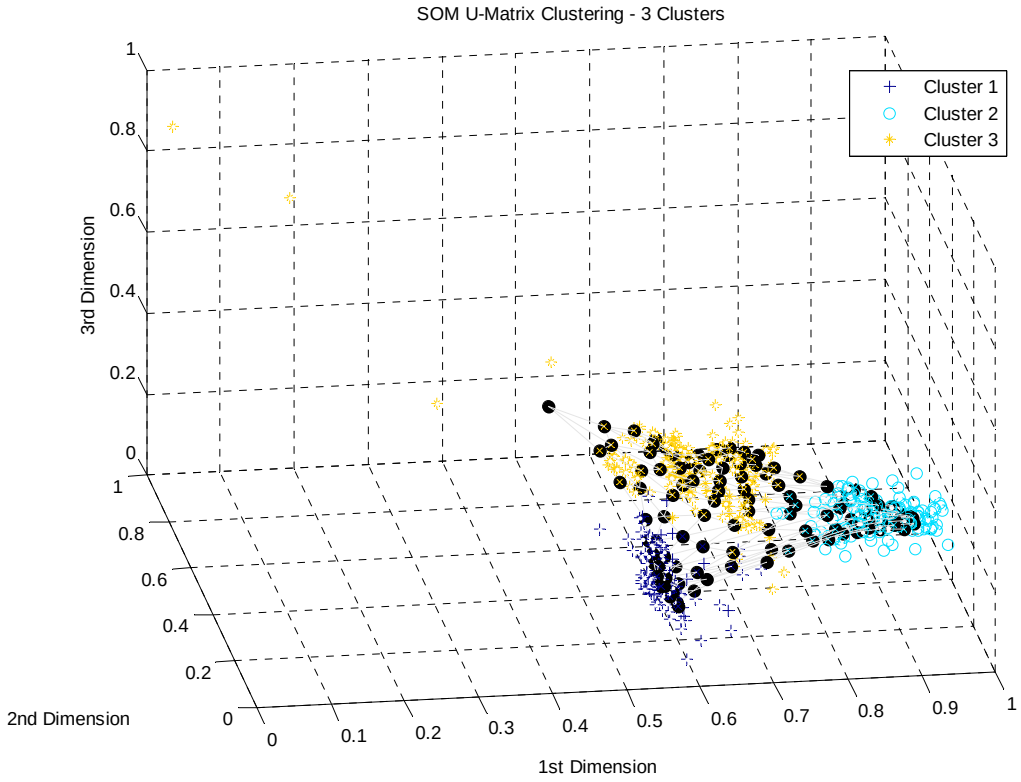
   Parameters described in *CLUSTERING* section.

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

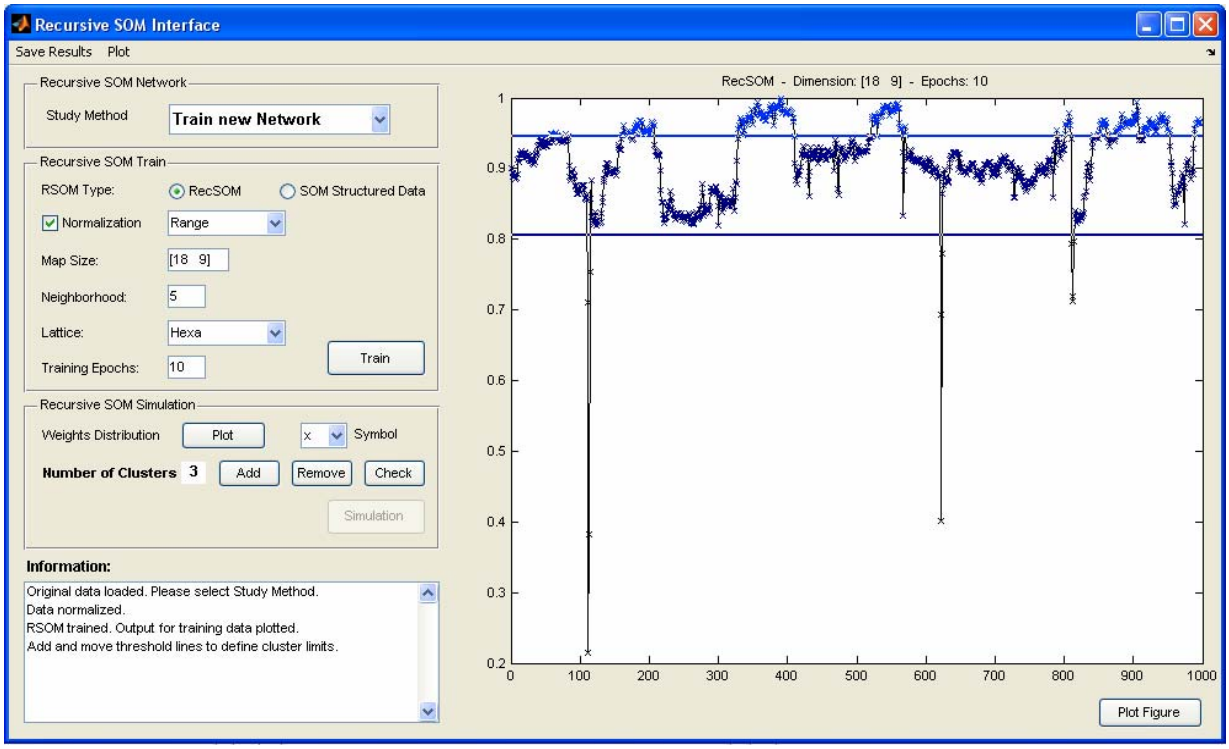
User can load *.mat* files containing SOM structures (*data structure* – sD – and *map structure* – sM) by clicking on *Load Structure* menu.

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[4].

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[5].

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.

[4] Thomas Voegtlin (2002), Recursive self-organizing maps, Neural Networks 15 (2002) 979–991

[5] Hagenbuchner, M, Sperduti, A & Tsoi, AC (2003), A self-organizing map for adaptive processing of structured data, IEEE Transactions on Neural Networks, May 2003, 14(3), 491-505

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.

Note: *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.

Note: *Plot* menu cotains 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.