DAta Mining & Exploration Program

Self-Organizing Feature Maps

User Manual

DAME-MAN-NA-0014

Issue: 1.1
Date: September 03, 2013
Author: M. Brescia, M. Guglielmo

Doc. : SOFM_UserManual_DAME-MAN-NA-0014-Rel1.1

INDEX

TABLE INDEX

FIGURE INDEX

1 Introduction

The present document is the user guide of the data mining models GSOM (Gated Self Organizing Maps) and CSOM (Clustering Self Organizing Maps), as implemented and integrated into the DAMEWARE. They are two customized versions of SOFM (Self Organizing Feature Maps) algorithm, dedicated to clustering functionality on FITS images (CSOM) and also on text files (GSOM). Optionally, embedded into the user parameters, there is also the possibility to execute hierarchical clustering, called MLC (Multi Layer Clustering), in which different clustering models can be layered to realize a multiple gated network.
This manual is one of the specific guides (one for each data mining model available in the webapp) having the main scope to help user to understand theoretical aspects of the model, to make decisions about its practical use in problem solving cases and to use it to perform experiments through the webapp, by also being able to select the right functionality associated to the model, based upon the specific problem and related data to be explored, to select the use cases, to configure internal parameters, to launch experiments and to evaluate results.

The documentation package consists also of a general reference manual on the webapp (useful also to understand what we intend for association between functionality and data mining model) and a GUI user guide, providing detailed description on how to use all GUI features and options.
So far, we strongly suggest to read these two manuals and to take a little bit of practical experience with the webapp interface before to explore specific model features, by reading this and the other model guides.
All the cited documentation package is available from the address
http://dame.dsf.unina.it/dameware.html **, where there is also the direct gateway to the beta webapp.**

As general suggestion, the only effort required to the end user is to have a bit of faith in Artificial Intelligence and a little amount of patience to learn basic principles of its models and strategies.

By merging for fun two famous commercial taglines we say: “*Think different, Just do it!*”
(casually this is an example of *data* (*text) mining*...!)

2 SOFM Model Theoretical Overview

This paragraph is intended to furnish a theoretical overview of the SOFM model (hereinafter terms SOFM and SOM will be considered synonyms), that is the reference machine learning algorithm whose custom implementation has been adopted in the CSOM e GSOM models plugged into the DAMEWARE web application.

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.
Main difference between SOM and SOFM models is that the latter requires to specify number of clusters to be found as input. It is a sort of driven training, where the output neuron layer try to distribute clusters in the resulting map. SOM original algorithm does not require such information and proceeds by itself with the cluster assignment.
This model stems from Kohonen [R5] and builds upon earlier work of Willshaw and von der Malsburg [R6]. The model is similar to the (much later developed) neural gas model since a decaying neighborhood range and adaptation strength are used. An important difference, however, is the topology which is constrained to be a two-dimensional grid and does not change during self-organization. This makes SOMs useful for visualizing low-dimensional views of high-dimensional data.

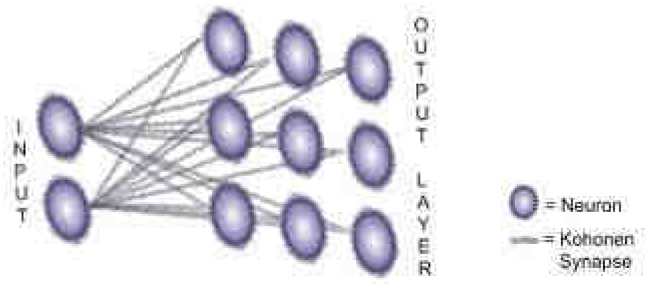


Fig. 1 – the typical SOFM/SOM network

Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. Mapping automatically classifies a new input vector.
A typical Kohonen SOM architecture consists of an input layer connected to a output layer (Two-dimensional Kohonen layer) via a Kohonen Connector consisting of Kohonen Synapses. Each neuron in a Kohonen Layer is associated with a unique set of co-ordinates in two-dimensional space, and hence is referred to as a Position Neuron. The input layer with 'n' input neurons is fed with n-dimensional input data one by one. The output layer organizes itself to represent the inputs. Hence the name 'self-organizing map'.

The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from data space onto the map is to first find the node with the closest weight vector to the vector taken from data space. Once the closest node is located it is assigned the values from the vector taken from the data space.

While it is typical to consider this type of network structure as related to feedforward networks where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation.
Useful extensions include using toroidal grids where opposite edges are connected and using large numbers of nodes. It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to K-means, larger self-organizing maps rearrange data in a way that is fundamentally topological in character.
The objective of SOM training is to ensure that different parts of the network respond similarly to similar input vectors. So, the training mainly involves analyzing the behavior of the network for a training sample and adjusting the weights of synapses to ensure that the network exhibits a similar behavior for a similar input. The training procedure involves following steps.

- *Initialize the weights to small random values;*
- *Choose a random training sample, assign the input vector to the input neurons and run the network;*
- *The output of a neuron in output layer represents the similarity between its weight vector (source synapse weights) and the input vector. The output neuron which has the highest output value is declared as the* ***winner neuron*** *for current input;*
- *Calculate the distance of each output neuron from the winner using a Neighborhood function;*
- *Update the weights of synapses using following formulae. (This adjusts the weights corresponding to winner and its neighbors such that, a neuron close to the current winner wins for a similar input):*
    - *Similarity = difference between source neurons output and the synapse weight;*
    - *Weight change = (learning_rate * neighborhood value of target neuron) * similarity;*
- *Similarly, train all samples in a random order. This completes one* ***training cycle*** *(or Training Epoch);*
- *Repeat the steps to complete the specified number of training epochs.*

The trained SOM maps any input vector to a winner neuron, which can be interpreted as the position of vector in two-dimensional space [R7].

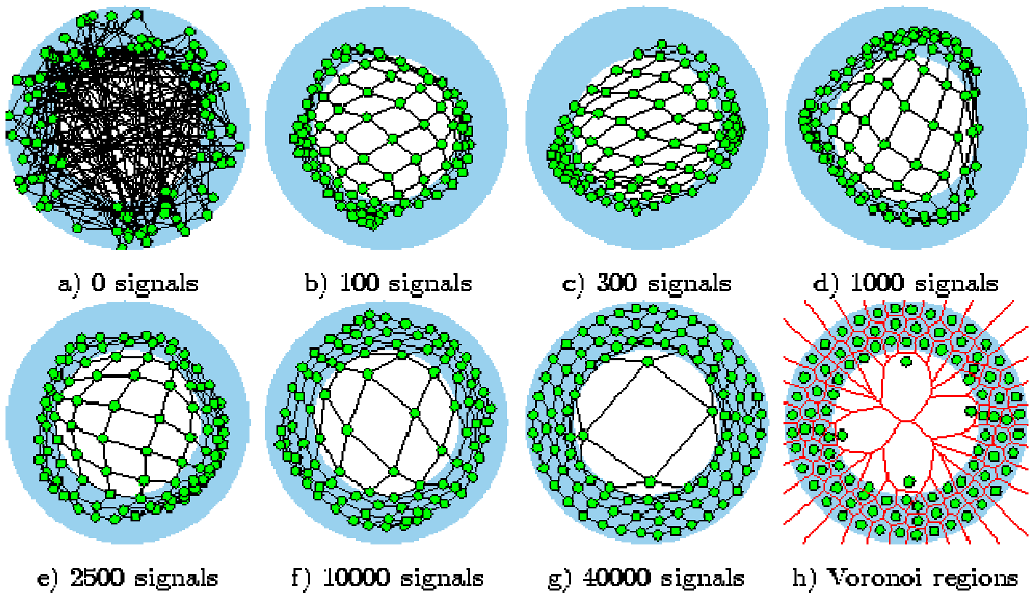


Fig. 2 – Example of a SOFM simulation for a simple ring-shaped data distribution [R7]

3 Use of the web application model

The SOM is one of the most common unsupervised neural architectures used in many application fields. It is especially related to clustering problems, and in DAME it is designed to be associated with such functionality domain. The description of this functionality is reported in the Reference Manual [A18], available from webapp menu or from the beta intro web page.

In the following are described practical information to configure the network architecture and the learning algorithm in order to launch and execute science cases and experiments.
There are two models available in the webapp. Both of them were extracted from an astronomical pipeline for image segmentation, called NEXT-II, designed and developed by DAME team (cfr. http://dame.dsf.unina.it/next.html). The modules extracted have been plugged into the DAMEWARE by using an internal software integration wrapping system and available as two clustering methods from the GUI.
The two models are:

- CSOM: specialized for clustering of FITS images using the tool NEXTII (library implemented in C++);
- GSOM: gated model for clustering on generic type of datasets (not only FITS images, but also FITS tabular or CSV data files);

3.1 Use Cases

For the user the two models, CSOM and GSOM, offer three use cases:

- ***Train***
- ***Test***
- ***Run***

3.2 Input

We also remark that massive datasets to be used in the various use cases are (and sometimes must be) different in terms of internal file content representation. Remind that in all DAME models it is possible to use one of the following data types:

- ASCII (extension .dat or .txt): simple text file containing rows (patterns) and columns (features) separated by spaces, normally without header;
- CSV (extension .csv): Comma Separated Values files, where columns are separated by commas;
- FITS (extension .fits): image or tabular fits files;
- VOTABLE (extension .votable): formatted files containing special fields separated by keywords coming from XML language, with more special keywords defined by VO data standards;

Depending on different use cases and experiments, the user should be able to perform a setup of many parameters and input/output files, without need to re-compile the code.
In order to implement this requirement, a set of input/output files has been designed.

3.3 GSOM I/O Files

3.3.1 Train Use Case

3.3.1.1 Input Files

As input to the program (depending on specific use case) the user must provide following setup files:

- **input dataset** (input data).

3.3.1.1.1 Input Dataset

The input dataset represents the input patterns to be processed for both training and/or run phases. These data must be submitted as ASCII, CSV, DAT TXT and more for GSOM model. The input file contains patterns with columns separated by spaces. Each pattern must be filled in as a row vector. All patterns must be of the same size.

3.3.1.2 Output Files

As output from the program, for GSOM training, provides following files:

- **GSOM_Train_weights.log** : this file contains the weights associated to the neurons of the net;
- **GSOM_Train_network_configuration.log**: this file is useful for run use case and contains all the parameter of configured network;

  - ✓ **row 1**: status boolean keys associated as follow: *configurationWait*, *configuring*, *firstTrainingWait*, *training* , *trained*;
  - ✓ **row 2**: values of *layersNumber* and *input pattern length*;
  - ✓ **row 3**: type of *ClusteringNN model* used;
  - ✓ **row 4**: status boolean keys associated as follow: *configurationWait*, *configuring*, *firstTrainingWait*, *training* , *trained*;
  - ✓ **row 5**: values of input pattern length and *number of clusters used* for train;
  - ✓ **row 6**: values of *initial and final learning rate* and *number of iterations*;

- ✓ **row 7**: values of *initial and final variance*, and *rows and columns* number of map of network;
- ✓ **the other row**: alternately, the file specifies the *pattern length* and the *weight vector* for each neuron of network;

- **GSOM_Train_status.log**: this file contains all parameters of configured experiments and error in case of failure;
- **GSOM_Train_params.xml**: this file contains all configuration parameters of experiment.

3.3.2 Run Use Case

3.3.2.1 Input Files

As input to the program the user must provide following setup files:

- **input dataset** (input data): the same type of file used on train case;
- **configuration file of network**: the output "GSOM_Train_network_configuration.log " obtained by train use case and added to the workspace of experiment.

3.3.2.2 Output Files

As output from GSOM **run** use case, the program provides following files:

- **GSOM_Run_weights.log** : this file contains the weights associated to the neurons of the net; the file settings are the same of train use case;
- **GSOM_Ruin_network_configuration.log**: this file is useful for run use case and contains all the parameter of configured network ; the file settings are the same of train use case;
- **GSOM_Run_status.log**: this file contains all parameters of experiment and error in case of failure; the file settings are the same of train use case;
- **GSOM_Run_ output_results.log**: this file contains the really output of the experiment:
  - ✓ **Column 1**: label of *winner cluster* for the input pattern;
  - ✓ **Column 2**: index of input pattern;
  - ✓ **Column 3**: input pattern length;
  - ✓ **Other columns:** values of input pattern;
- **GSOM_Run_plot_clusters.jpeg**: image obtained by "GSOM_Run_output_results.log";
  GSOM_Run_clusters_count.log: this file contains, for each cluster, the number of pattern associated during the clustering and the percentage of association on respect to total number of patterns;
  - ✓ **Column 1** : label of cluster;
  - ✓ **Column 2** : total number of pattern associated for the cluster;
  - ✓ **Column 3** : percentage of association;
- **GSOM_Run_params.xml**: this file contains all configuration parameters for the experiment;

3.3.3 Test Use Case

3.3.3.1 Input Files

As input to the program the user must provide following setup files:

- **input dataset** (input data): the same type of file used on train case;
- **configuration file of network**: the output "GSOM_Train_network_configuration.log" or "GSOM_Run_network_configuration.log" obtained by train or run use case and added to the workspace of experiment.

3.3.3.2 Output Files

As output from GSOM test use case, the program provides following files:

- **GSOM_Test_network_configuration.log**: this file is useful for run use case and contains all the parameter of configured network ; the file settings are the same of train use case;

- **GSOM_Test_status.log**: this file contains all parameters of experiment and error in case of failure; the file settings are the same of train use case;
- **GSOM_Test_weights_I_dataset.log** : this file contains the weights associated to the neurons of the net used for the first dataset; the file settings are the same of train use case;
- **GSOM_Test_weights_II_dataset.log** : this file contains the weights associated to the neurons of the net used for the second dataset; the file settings are the same of train use case;
- **GSOM_Test__output_results_I_dataset.log**: this file contains the output of the experiment on the first dataset;
- **GSOM_Test__output_results_II_dataset.log**: this file contains the output of the experiment on the second dataset;
- **GSOM_Test_plot_clusters_I_dataset.jpeg**: image obtained by "GSOM_Run_output_results_I_dataset.log
- **GSOM_Test_plot_clusters_II_dataset.jpeg**: image obtained by "GSOM_Run_output_results_II_dataset.log";
- **GSOM_Test_clusters_count_I_dataset.log**: this file contains, for each cluster, the number of pattern associated during the clustering and the percentage of association on respect to total number of patterns of first dataset;
  - ✓ **Column 1** : label of cluster;
  - ✓ **Column 2** : total number of pattern associated for the cluster;
  - ✓ **Column 3** : percentage of association;
- **GSOM_Test_clusters_count_II_dataset.log**: this file contains, for each cluster, the number of pattern associated during the clustering and the percentage of association on respect to total number of patterns of second dataset;
  - ✓ **Column 1** : label of cluster;
  - ✓ **Column 2** : total number of pattern associated for the cluster;
  - ✓ **Column 3** : percentage of association;
- **GSOM_Test_overlap_dataset.dat**: this file contains patterns of overlapping between first dataset and second dataset; the two dataset are obtained by splitting the input dataset;

- **GSOM_statistical_output**: this file contains the following results:

  - ✓ **Column 1:** Label of cluster;
  - ✓ **Column 2:** occurrence of cluster on clustering first dataset;
  - ✓ **Column 3:** occurrence of cluster on clustering second dataset;
  - ✓ **Column 4:** percentage of association.

3.4 CSOM I/O Files

3.4.1 Train Use Case

3.4.1.1 Input Files

As input to the program (depending on specific use case) the user must provide following setup files:

- **input dataset** (input data).

3.4.1.1.1 Input Dataset

The input dataset represents the input patterns to be processed for both training and/or run phases. These data must be submitted FITS for CSOM model. The input file contains a fits astronomical image or table.

3.4.1.2 Output Files

As output from the program, for CSOM training, provides following files:

- **CSOM_Train_weights.log** : this file contains the weights associated to the neurons of the net;

- **CSOM_Train_network_configuration.log**: this file is useful for run use case and contains all the parameter of configured network;

  - ✓ **row 1**: status boolean keys associated as follow: *configurationWait*, *configuring*, *firstTrainingWait*, *training* , *trained*;
  - ✓ **row 2**: values of *layersNumber* and *input pattern length*;
  - ✓ **row 3**: type of *ClusteringNN model* used;
  - ✓ **row 4**: status boolean keys associated as follow: *configurationWait*, *configuring*, *firstTrainingWait*, *training* , *trained*;
  - ✓ **row 5**: values of input pattern length and *number of clusters used* for train;
  - ✓ **row 6**: values of *initial and final learning rate* and *number of iterations*;
  - ✓ **row 7**: values of *initial and final variance*, and *rows and columns* number of map of network;
  - ✓ **the other row**: alternately, the file specifies the *pattern length* and the *weight vector* for each neuron of network;

- **CSOM_Train_status.log**: this file contains all parameters of configured experiments and error in case of failure;
- **CSOM_Train_params.xml**: this file contains all configuration parameters of experiment.

3.4.2 Run Use Case

3.4.2.1 Input Files

As input to the program the user must provide following setup files:

- **input dataset** (input data): the same type of file used on train case;
- **configuration file of network**: the output "CSOM_Train_network_configuration.log " obtained by train use case and added to the workspace of experiment.

3.4.2.2 Output Files

As output from CSOM **run** use case, the program provides following files:

- **GSOM_Run_weights.log** : this file contains the weights associated to the neurons of the net; the file settings are the same of train use case;
- **CSOM_Ruin_network_configuration.log**: this file is useful for run use case and contains all the parameter of configured network ; the file settings are the same of train use case;
- **CSOM_Run_status.log**: this file contains all parameters of experiment and error in case of failure; the file settings are the same of train use case;
- **CSOM_Run_ output_results.log**: this file contains the really output of the experiment:
  - ✓ **Column 1**: label of *winner cluster* for the input pattern;
  - ✓ **Column 2**: index of input pattern;
  - ✓ **Column 3**: input pattern length;
  - ✓ **Other columns:** values of input pattern;
- **CSOM_Run_plot_clusters.jpeg**: image obtained by "CSOM_Run_output_results.log";
- **CSOM_Run_clusters_count.log**: this file contains, for each cluster, the number of pattern associated during the clustering and the percentage of association on respect to total number of patterns;
  - ✓ **Column 1** : label of cluster;
  - ✓ **Column 2** : total number of pattern associated for the cluster;
  - ✓ **Column 3** : percentage of association;
- **CSOM_Run_params.xml**: this file contains all configuration parameters for the experiment;

3.4.3 Test Use Case

3.4.3.1 Input Files

As input to the program the user must provide following setup files:

- **input dataset** (input data): the same type of file used on train case;
- **configuration file of network**: the output "CSOM_Train_network_configuration.log" or "CSOM_Run_network_configuration.log" obtained by train or run use case and added to the workspace of experiment.

3.4.3.2 Output Files

As output from CSOM test use case, the program provides following files:

- **CSOM_Test_network_configuration.log**: this file is useful for run use case and contains all the parameter of configured network ; the file settings are the same of train use case;
- **CSOM_Test_status.log**: this file contains all parameters of experiment and error in case of failure; the file settings are the same of train use case;
- **CSOM_Test_weights_I_dataset.log** : this file contains the weights associated to the neurons of the net used for the first dataset; the file settings are the same of train use case;
- **CSOM_Test_weights_II_dataset.log** : this file contains the weights associated to the neurons of the net used for the second dataset; the file settings are the same of train use case;
- **CSOM_Test__output_results_I_dataset.log**: this file contains the output of the experiment on the first dataset;
- **CSOM_Test__output_results_II_dataset.log**: this file contains the output of the experiment on the second dataset;
- **CSOM_Test_plot_clusters_I_dataset.jpeg**: image obtained by "CSOM_Run_output_results_I_dataset.log;
- **CSOM_Test_plot_clusters_II_dataset.jpeg**: image obtained by "CSOM_Run_output_results_II_dataset.log";
- **CSOM_Test_clusters_count_I_dataset.log**: this file contains, for each cluster, the number of pattern associated during the clustering and the percentage of association on respect to total number of patterns of first dataset;
  - ✓ **Column 1** : label of cluster;
  - ✓ **Column 2** : total number of pattern associated for the cluster;
  - ✓ **Column 3** : percentage of association;
- **CSOM_Test_clusters_count_II_dataset.log**: this file contains, for each cluster, the number of pattern associated during the clustering and the percentage of association on respect to total number of patterns of second dataset;
  - ✓ **Column 1** : label of cluster;
  - ✓ **Column 2** : total number of pattern associated for the cluster;
  - ✓ **Column 3** : percentage of association;
- **CSOM_Test_overlap_dataset.dat**: this file contains patterns of overlapping between first dataset and second dataset; the two dataset are obtained by splitting the input dataset;
- **CSOM_statistical_output.log**: this file contains the following results:

  - ✓ **Column 1:** Label of cluster;
  - ✓ **Column 2:** occurrence of cluster on clustering first dataset;
  - ✓ **Column 3:** occurrence of cluster on clustering second dataset;
  - ✓ **Column 4:** percentage of association.

3.5 Output files table of CSOM and GSOM models

Use Case	Output Files CSOM Model
TRAIN	CSOM_Train_status.log
	CSOM_Train_weigths.log
	CSOM_ Train_network_configuration.log
RUN	CSOM_Run_ status.log
	CSOM_ Run_ network_configuration.log
	CSOM_ Run_ weigths.log
	CSOM_ Run_output_results.log
	CSOM_ Run_clusters_count.log
	CSOM_ Run_plot_clusters.jpeg
TEST	CSOM_Test_ status.log
	CSOM_ Test_ network_configuration.log
	CSOM_Test_statistical_output.log
	CSOM_Test_overlap_dataset.dat
	CSOM_ Test_ weigths_I_dataset.log
	CSOM_ Test_ weigths_II_dataset.log
	CSOM_ Test_ output_results_I_dataset.log
	CSOM_ Test_ output_results_II_dataset.log
	CSOM_ Test_ clusters_count_I_dataset.log
	CSOM_ Test_ clusters_count_II_dataset.log
	CSOM_ Test_ plot_clusters_I_dataset.jpeg
	CSOM_ Test_ plot_clusters_II_dataset.jpeg

Tab. 1 – CSOM model output files table

Use Case	Output Files GSOM Model
TRAIN	GSOM_Train_status.log
	GSOM_Train_weigths.log
	GSOM_ Train_network_configuration.log
RUN	GSOM_Run_ status.log
	GSOM_ Run_ network_configuration.log
	GSOM_ Run_ weigths.log
	GSOM_ Run_output_results.log
	GSOM_ Run_clusters_count.log
	GSOM_ Run_plot_clusters.jpeg
TEST	GSOM_Test_ status.log
	GSOM_ Test_ network_configuration.log
	GSOM_Test_statistical_output.log
	GSOM_Test_overlap_dataset.dat
	GSOM_ Test_ weigths_I_dataset.log
	GSOM_ Test_ weigths_II_dataset.log
	GSOM_ Test_ output_results_I_dataset.log
	GSOM_ Test_ output_results_II_dataset.log
	GSOM_ Test_ clusters_count_I_dataset.log
	GSOM_ Test_ clusters_count_II_dataset.log
	GSOM_ Test_ plot_clusters_I_dataset.jpeg
	GSOM_ Test_ plot_clusters_II_dataset.jpeg

Tab. 2 – GSOM model output files table

3.6 TRAIN Use case

In the use case named "**Train**", the software provides the possibility to train the SOFM model of NEXTII library. The user will be able to set training parameters, set training dataset, manipulate the training dataset and execute the training experiments.

There are several parameters to be set to achieve training, dealing with network topology and learning algorithm. In the experiment configuration there is also the Help button, redirecting to a web page dedicated to support the user with deep information about all parameters and their default values.

We remark that all parameters labeled by an asterisk are considered required. In all other cases the fields can be left empty (default values are used).

3.6.1 Clustering with CSOM model – Train Parameter Specifications

In the case of Clustering_CSOM with Train use case, the help page is at the address:
http://dame.dsf.unina.it/clustering_csom.html#class_train

- **Input Dataset File**

  This parameter is a field required!

  This is the dataset file to be used as input for the learning phase of the model. It typically must not include target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must FITS-IMAGE.

- **Number of clusters/neurons (I layer)**

  This parameter is a field required!

  This is the number of neurons of the first layer of the Multi Layer network. As suggestion this should be selected as an integer equal to or greater than two neurons. **Note that number of neurons corresponds to the number of searched clusters.**

- **Number of Iterations**

  This is the number of training iterations. As suggestion this should be selected as an integer greater than zero.

  If left empty, its default value is 1000.

- **Diameter**

  It is the pixel neighborhood diameter, As suggestion this should be selected as an integer greater than two.

If left empty, its default value is 3.

- **Number of Layers**

  This is the number of layers in case of choice of hierarchical Multi layer clustering model. As suggestion this should be selected into integer range between 1 and 3.

  If left empty, its default value is 1 (i.e. no multi layer clustering, but simple SOFM model).

- **Initial variance (I layer)**

  This is the initial variance of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as real number greater than 0 used on neighborhood function.

  If left empty, its default value is 4.0.

- **Final variance (I layer)**

  This is the final variance of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as Real number between 0 and value of initial variance of the first layer.

  If left empty, its value default is 0.001.

- **Initial learning rate (I layer)**

  This is the initial learning rate of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as real number between 0 and 1.

  If left empty, its default value is 0.7.

- **Final learning rate (I layer)**

  This is the final learning rate of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as real number between 0 and 1 (including 0) **STRICTLY LESS** than value of initial learning rate of the first layer.

  If left empty, its default value is 0.005.

- **Number of clusters/neurons (II Layer)**

  This is the number of clusters/Neurons of second Layer, specified only in the case of Multi layer clustering.

As suggestion this should be selected as integer number greater than 2 less than number of neurons of first layer and greater than number of neurons of third layer (if specified).

However, this number is set to default value in the two following case:

1. the number of layers is equal to 2 and number of neurons of second layer is not specified: in this case, its default value is 8;
2. the number of layers is equal to 2 and number of neurons of second layer is greater than number specified for first layer: in this case, there are two solutions:
   a. if number neurons of first layer is STRICTLY greater than 2, number of neurons of second layer is obtained by decreasing number of neurons in the first layer;
   b. if number neurons of first layer is equal to 2 (minimum required), uses the following default values:
      i. number neurons of first layer : 10;
      ii. number neurons of second layer : 8.

- **Initial variance (II Layer)**

  This is the initial variance of first layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number greater than 0 used on neighborhood function.

  If left empty, its default value is 3.0.

- **Final variance (II Layer)**

  This is the final variance of second layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and value of initial variance of the second layer.

  If left empty, its default value is 0.001.

- **Initial learning rate (II Layer)**

  This is the initial learning rate of second layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1.

  If left empty, its default value is 0.7.

- **Final learning rate (II Layer)**

  This is the final learning rate of second layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1 (including 0) **STRICTLY LESS** than value of initial learning rate of the second layer.

If left empty, its default value is 0.005.

- **Number of clusters/neurons (III layer)**

  This is the number of clusters/Neurons of third Layer, specified only in the case of Multi layer clustering.

  As suggestion this should be selected as integer number greater than 2 less than number of neurons of second layer.

  However, this number is set to default value in the two following case:

  1. the number of layers is equal to 3 and number of third layer is not specified: in this case, its default value is 4;
  2. the number of layers is equal to 3 and <u>number of neurons of third layer is greater than number specified for second layer</u> or <u>number of neurons of third layer is greater than number specified for first layer</u> or <u>number of neurons of second layer is greater than number specified for first layer</u> : in this case, there are two solutions:
     a. if number neurons of first layer is STRICTLY greater than 4, number of neurons of second layer is obtained by decreasing number of neurons in the first layer and number of neurons of third layer is obtained by decreasing number of neurons in second layer;
     b. if number neurons of first layer is equal to 2 (minimum required), uses the following default values:
        i. number neurons of first layer : 10;
        ii. number neurons of second layer : 8;
        iii. number neurons of third layer : 4.

- **Initial variance (III Layer)**

  This is the initial variance of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number greater than 0 used on neighborhood function.

  If left empty, its default value is 2.0.

- **Final variance (III Layer)**

  This is the final variance of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and value of initial variance of the third layer.

  If left empty, its default value is 0.001.

- **Initial learning rate (III Layer)**

  This is the initial learning rate of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1.

  If left empty, its default value is 0.7.

- **Final learning rate (III Layer)**

  This is the final learning rate of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1 (including 0) **STRICTLY LESS** than value of initial learning rate of the third layer.

  If left empty, its default value is 0.005.

3.6.2 Clustering with GSOM – Train Parameter Specifications

In the case of Clustering_GSOM with Train use case, the help page is at the address: http://dame.dsf.unina.it/clustering_gsom.html#class_train

- **Input Dataset File**

  This parameter is a field required!

  This is the dataset file to be used as input for the learning phase of the model. It typically must not include target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be FITS-TABLE, ASCII and CSV.

- **Number of clusters/neurons (I layer)**

  This parameter is a field required!

  This is the number of neurons of the first layer of the Multi Layer network. As suggestion this should be selected as an integer equal to or greater than two neurons. **Note that number of neurons corresponds to the number of searched clusters.**

- **Number of Iterations**

  This is the number of training iterations. As suggestion this should be selected as an integer greater than zero.

  If left empty, its default value is 1000.

- **Diameter**

  It is the pixel neighborhood diameter, As suggestion this should be selected as an integer greater than two.

  If left empty, its default value is 3.

- **Number of Layers**

  This is the number of layers in case of choice of hierarchical Multi layer clustering model. As suggestion this should be selected into integer range between 1 and 3.

  If left empty, its default value is 1 (i.e. no multi layer clustering, but simple SOFM model).

- **Initial variance (I layer)**

  This is the initial variance of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as real number greater than 0 used on neighborhood function.

  If left empty, its default value is 4.0.

- **Final variance (I layer)**

  This is the final variance of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as Real number between 0 and value of initial variance of the first layer.

  If left empty, its value default is 0.001.

- **Initial learning rate (I layer)**

  This is the initial learning rate of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as real number between 0 and 1.

  If left empty, its default value is 0.7.

- **Final learning rate (I layer)**

  This is the final learning rate of first layer of hierarchical Multi layer clustering model. As suggestion this should be selected as real number between 0 and 1 (including 0) **STRICTLY LESS** than value of initial learning rate of the first layer.

  If left empty, its default value is 0.005.

- **Number of clusters/neurons (II Layer)**

  This is the number of clusters/Neurons of second Layer, specified only in the case of Multi layer clustering.

  As suggestion this should be selected as integer number greater than 2 less than number of neurons of first layer and greater than number of neurons of third layer (if specified).

  However, this number is set to default value in the two following case:

  1. the number of layers is equal to 2 and number of neurons of second layer is not specified: in this case, its default value is 8;
  2. the number of layers is equal to 2 and number of neurons of second layer is greater than number specified for first layer: in this case, there are two solutions:
     a. if number neurons of first layer is STRICTLY greater than 2, number of neurons of second layer is obtained by decreasing number of neurons in the first layer;
     b. if number neurons of first layer is equal to 2 (minimum required), uses the following default values:
        i. number neurons of first layer : 10;
        ii. number neurons of second layer : 8.

- **Initial variance (II Layer)**

  This is the initial variance of first layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number greater than 0 used on neighborhood function.

  If left empty, its default value is 3.0.

- **Final variance (II Layer)**

  This is the final variance of second layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and value of initial variance of the second layer.

  If left empty, its default value is 0.001.

- **Initial learning rate (II Layer)**

  This is the initial learning rate of second layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1.

  If left empty, its default value is 0.7.

- **Final learning rate (II Layer)**

  This is the final learning rate of second layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1 (including 0) **STRICTLY LESS** than value of initial learning rate of the second layer.

  If left empty, its default value is 0.005.

- **Number of clusters/neurons (III layer)**

  This is the number of clusters/Neurons of third Layer, specified only in the case of Multi layer clustering.

  As suggestion this should be selected as integer number greater than 2 less than number of neurons of second layer.

  However, this number is set to default value in the two following case:

  1. the number of layers is equal to 3 and number of third layer is not specified: in this case, its default value is 4;
  2. the number of layers is equal to 3 and number of neurons of third layer is greater than number specified for second layer or number of neurons of third layer is greater than number specified for first layer or number of neurons of second layer is greater than number specified for first layer : in this case, there are two solutions:
     a. if number neurons of first layer is STRICTLY greater than 4, number of neurons of second layer is obtained by decreasing number of neurons in the first layer and number of neurons of third layer is obtained by decreasing number of neurons in second layer;
     b. if number neurons of first layer is equal to 2 (minimum required), uses the following default values:
        i. number neurons of first layer : 10;
        ii. number neurons of second layer : 8;
        iii. number neurons of third layer : 4.

- **Initial variance (III Layer)**

  This is the initial variance of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number greater than 0 used on neighborhood function.

  If left empty, its default value is 2.0.

- **Final variance (III Layer)**

  This is the final variance of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and value of initial variance of the third layer.

  If left empty, its default value is 0.001.

- **Initial learning rate (III Layer)**

  This is the initial learning rate of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1.

  If left empty, its default value is 0.7.

- **Final learning rate (III Layer)**

  This is the final learning rate of third layer, specified only in the case of Multi layer clustering. As suggestion this should be selected as real number between 0 and 1 (including 0) **STRICTLY LESS** than value of initial learning rate of the third layer.

  If left empty, its default value is 0.005.

3.7 TEST Use case

In the use case named "**Test**", the software provides the possibility to test the stability of SOFM model. As known, in the unsupervised case, it cannot be provided the training validation, in the same way as intended and formalized in the supervised case, but a statistical evaluation about the stability of clustering in different training sessions can be approached, i.e. by collecting two separate training datasets, having a chosen overlap percentage and comparing the training results between them. This use case is offered as an option to the training use case. If the user select it, he may choose to add a column to the original dataset. This column will have as label the train set cluster membership.

The user will be able to use already trained CSOM and GSOM models, their weight configurations to execute the testing experiments.

In the experiment configuration there is also the Help button, redirecting to a web page dedicated to support the user with deep information about all parameters and their default values.

We remark that all parameters labeled by an asterisk are considered required. In all other cases the fields can be left empty (default values are used).

3.7.1 Clustering with CSOM – Test Parameter Specifications

In the case of Clustering_CSOM with Test use case, the help page is at the address:

http://dame.dsf.unina.it/clustering_csom.html#class_test

- **Input File Dataset**

  This parameter is a field required!

  This is the dataset file to be used as input for the testing. It typically must not include target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must FITS-IMAGE. For example, it could be the same dataset file used as the training input file.

- **Configuration file of trained net**

  This parameter is a field required!

  This is a file generated by the model during training phase. It contains the resulting network topology as stored at the end of a training session. Usually this file should not be edited or modified by users, just to preserve its content as generated by the model itself.

- **Splitting percentage of input dataset**

  This is a real number representing the splitting percentage of input dataset. As suggestion this should be selected in a range between 1 and 100.

  If left empty, its default value is 65.0 (%).

3.7.2 Clustering with GSOM – Test Parameter Specifications

In the case of Classification_GSOM with Test use case, the help page is at the address:
http://dame.dsf.unina.it/clustering_gsom.html#class_test

- **Input File Dataset**

  This parameter is a field required!

  This is the dataset file to be used as input for the learning phase of the model. It typically must not include target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be FITS-TABLE, ASCII and CSV

- **Configuration file of trained net**

  This parameter is a field required!

  This is a file generated by the model during training phase. It contains the resulting network topology as stored at the end of a training session. Usually this file should not be edited or modified by users, just to preserve its content as generated by the model itself.

- **Splitting percentage of input dataset**

  This is a real number representing the splitting percentage of input dataset. As suggestion this should be selected in a range between 1 and 100.

  If left empty, its default value is 65.0 (%).

3.8 RUN Use case

In the use case named "**Run**", the software provides the possibility to execute clustering with SOFM models. The user will be able to use already trained CSOM and GSOM models to execute the normal experiments.

In the experiment configuration there is also the Help button, redirecting to a web page dedicated to support the user with deep information about all parameters and their default values.

We remark that all parameters labeled by an asterisk are considered required. In all other cases the fields can be left empty (default values are used).

3.8.1 Clustering with CSOM – Run Parameter Specifications

In the case of Clustering_CSOM with Run use case, the help page is at the address: http://dame.dsf.unina.it/clustering_csom.html#class_run

- **Input File Dataset**

  This parameter is a field required!

  This is the dataset file to be used as input for the testing. It typically must not include target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must FITS-IMAGE. For example, it could be the same dataset file used as the training input file.

- **Configuration file of trained net**

  This parameter is a field required!

  This is a file generated by the model during training phase. It contains the resulting network topology as stored at the end of a training session. Usually this file should not be edited or modified by users, just to preserve its content as generated by the model itself.

3.8.2 Clustering with GSOM – Run Parameter Specifications

In the case of Clustering_GSOM with Run use case, the help page is at the address: http://dame.dsf.unina.it/clustering_gsom.html#class_run

- **Input File Dataset**

  This parameter is a field required!

  This is the dataset file to be used as input for the learning phase of the model. It typically must not include target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be FITS-TABLE, ASCII and CSV

- **Configuration file of trained net**

  This parameter is a field required!

  This is a file generated by the model during training phase. It contains the resulting network topology as stored at the end of a training session. Usually this file should not be edited or modified by users, just to preserve its content as generated by the model itself

4 Examples

This section is dedicated to show some practical examples of the correct use of the web application.
Not all aspects and available options are reported, but a significant sample of features useful for beginners of DAME suite and with a poor experience about data mining methodologies with machine learning algorithms.
In order to do so, very simple and trivial problems will be described.
Further complex examples will be integrated here in the next releases of the documentation.

4.1 Usage Example of Clustering_GSOM

The problem can be stated as follows: we want to execute experiment on WINE.CSV file[1]. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The attributes are :

- Alcohol
- Malic acid
- Ash
- Alcalinity of ash
- Magnesium
- Total phenols
- Flavanoids
- Nonflavanoid phenols
- Proanthocyanins
- Color intensity
- Hue
- OD280/OD315 of diluted wines
- Proline

[1] To see more information about wine.csv dataset, see the link: http://archive.ics.uci.edu/ml/datasets/Wine

All attributes are continuous but NOTE that first column on wine dataset is a class identifier (1-3). In other words each class identify a cultivars.
Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data.
A *cluster* is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters. In this case, the clusters can be associated to the different cultivars and so, the problem is to associate the pattern of wine dataset to this clusters.
The starting point of experiment is to create a new workspace, named **MLCSOMExp**, to populate it with the file **wine.csv**: CSV dataset file used on GSOM experiment.

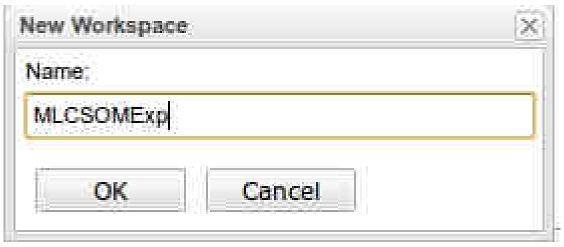
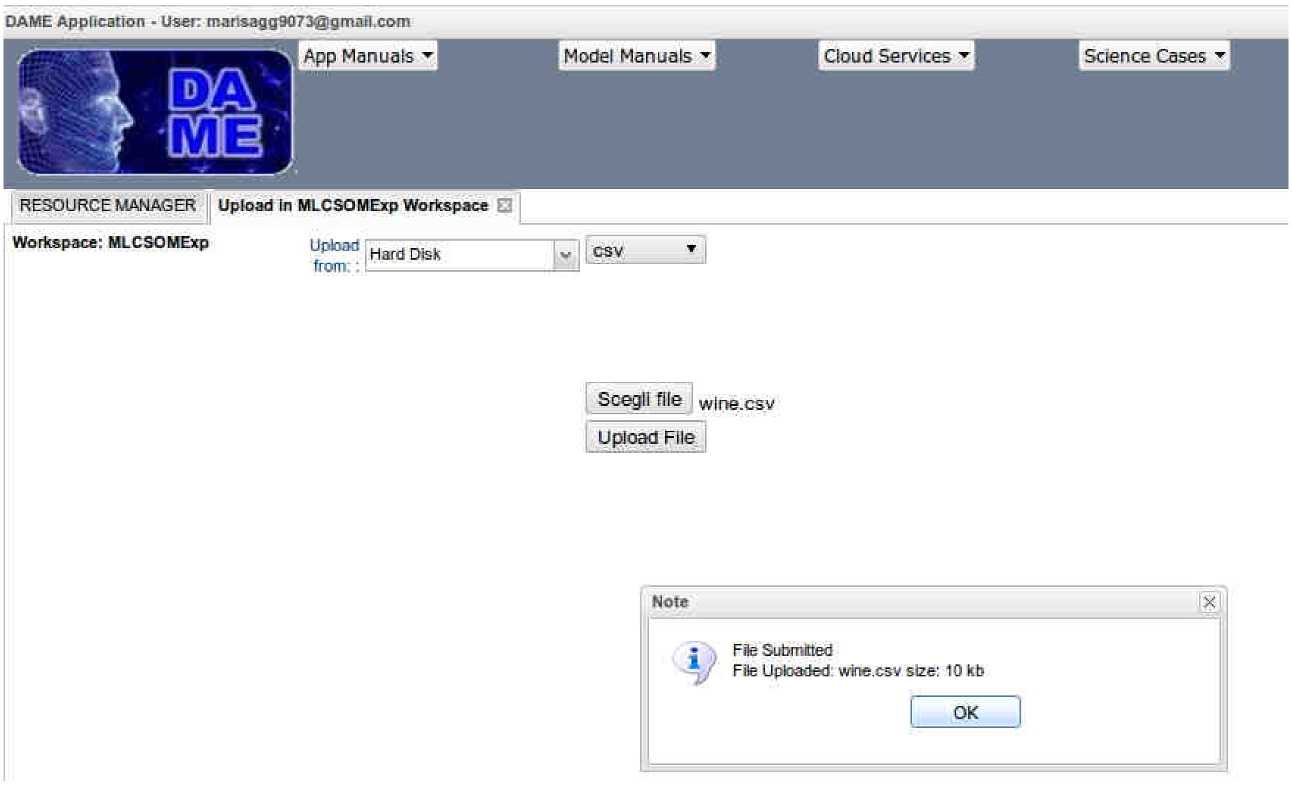


Fig. 3 – The starting point, creation tab of New Workspace

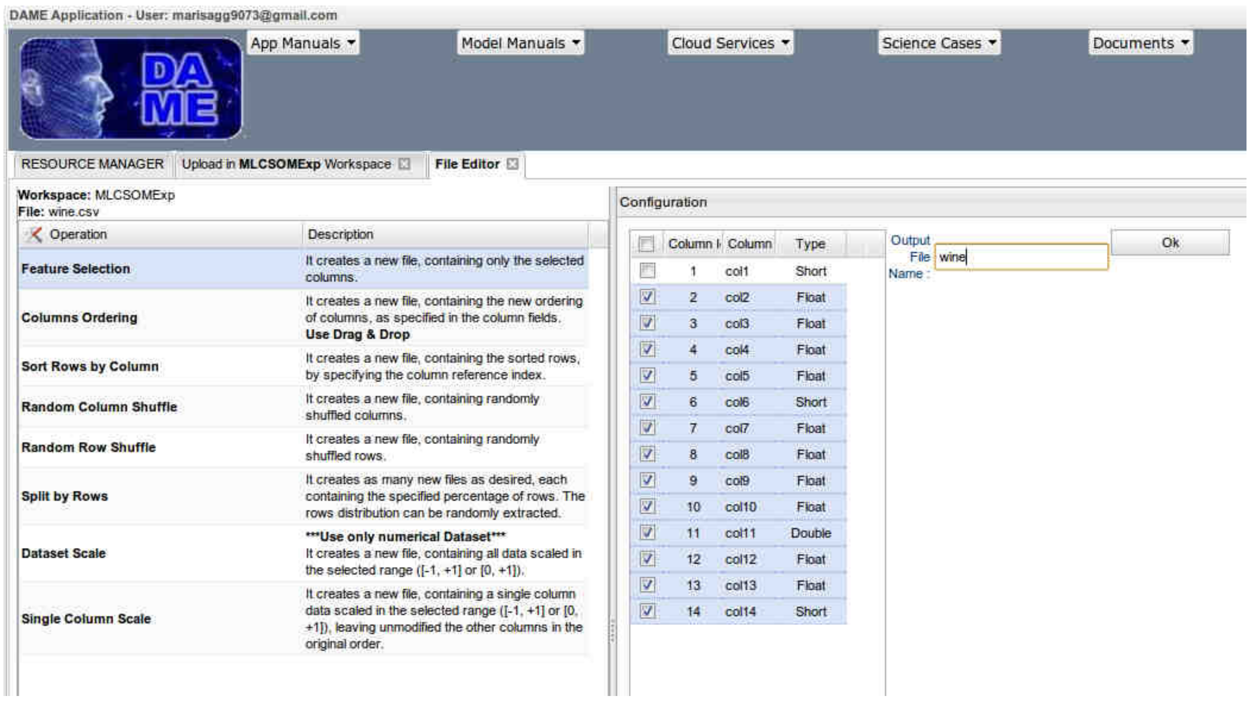
First of all, after uploading of the file, you need to make the wine dataset useful in a clustering experiment.



File Submitted
File Uploaded: wine.csv size: 10 kb

Fig. 4 – Upload Manager Tab: uploading wine.csv file

So, you must delete the first "target" column. This can be done through the application using the **File Editor**, as shown:



Workspace: MLCSOMExp
File: wine.csv

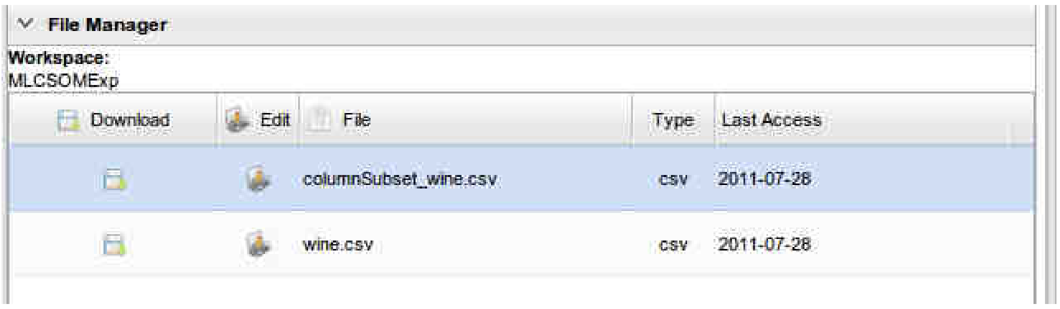
Operation	Description
Feature Selection	It creates a new file, containing only the selected columns.
Columns Ordering	It creates a new file, containing the new ordering of columns, as specified in the column fields. **Use Drag & Drop**
Sort Rows by Column	It creates a new file, containing the sorted rows, by specifying the column reference index.
Random Column Shuffle	It creates a new file, containing randomly shuffled columns.
Random Row Shuffle	It creates a new file, containing randomly shuffled rows.
Split by Rows	It creates as many new files as desired, each containing the specified percentage of rows. The rows distribution can be randomly extracted.
Dataset Scale	*****Use only numerical Dataset***** It creates a new file, containing all data scaled in the selected range ([-1, +1] or [0, +1]).
Single Column Scale	It creates a new file, containing a single column data scaled in the selected range ([-1, +1] or [0, +1]), leaving unmodified the other columns in the original order.

	Column I	Column	Type
☐	1	col1	Short
☑	2	col2	Float
☑	3	col3	Float
☑	4	col4	Float
☑	5	col5	Float
☑	6	col6	Short
☑	7	col7	Float
☑	8	col8	Float
☑	9	col9	Float
☑	10	col10	Float
☑	11	col11	Double
☑	12	col12	Float
☑	13	col13	Float
☑	14	col14	Short

Output File Name : wine

Fig. 5 – File Editor Tab

You must select useful columns, name for the new file and save it. The new name file will be: columnSubset_<name>.<format-file>. So, in this case we have a new file named **columnSubset_wine.csv**.



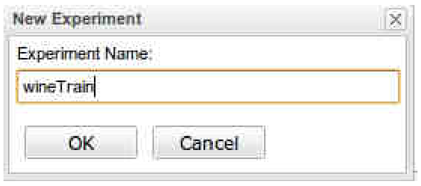
Workspace: MLCSOMExp

Download	Edit	File	Type	Last Access
		columnSubset_wine.csv	csv	2011-07-28
		wine.csv	csv	2011-07-28

Fig. 6 – File Manager: after editing wine.csv file

4.1.1 Clustering_GSOM Train Use Case

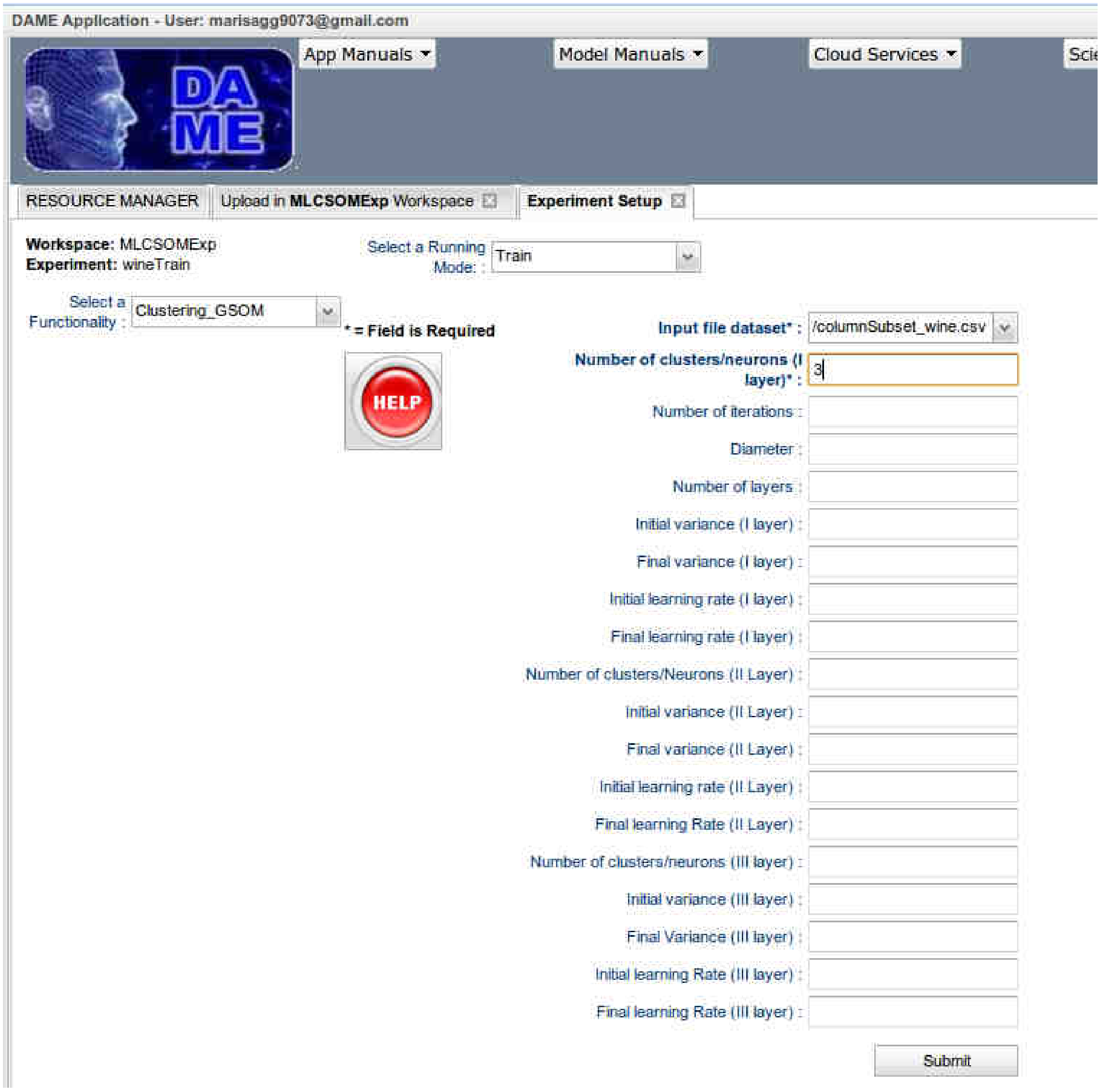
Let suppose we create an experiment, named **wineTrain**, and we want configure it.



Experiment Name: wineTrain

Fig. 7 – The wineTrain experiment creation tab

After creation, the new configuration tab is open. Here we select Clustering_GSOM as couple functionality-model of the current experiment and we select also Train use case.



Workspace: MLCSOMExp
Experiment: wineTrain

Select a Running Mode: Train

Select a Functionality: Clustering_GSOM

* = Field is Required

Input file dataset* :	/columnSubset_wine.csv
Number of clusters/neurons (I layer)* :	3
Number of iterations :	
Diameter :	
Number of layers :	
Initial variance (I layer) :	
Final variance (I layer) :	
Initial learning rate (I layer) :	
Final learning rate (I layer) :	
Number of clusters/Neurons (II Layer) :	
Initial variance (II Layer) :	
Final variance (II Layer) :	
Initial learning rate (II Layer) :	
Final learning Rate (II Layer) :	
Number of clusters/neurons (III layer) :	
Initial variance (III layer) :	
Final Variance (III layer) :	
Initial learning Rate (III layer) :	
Final learning Rate (III layer) :	

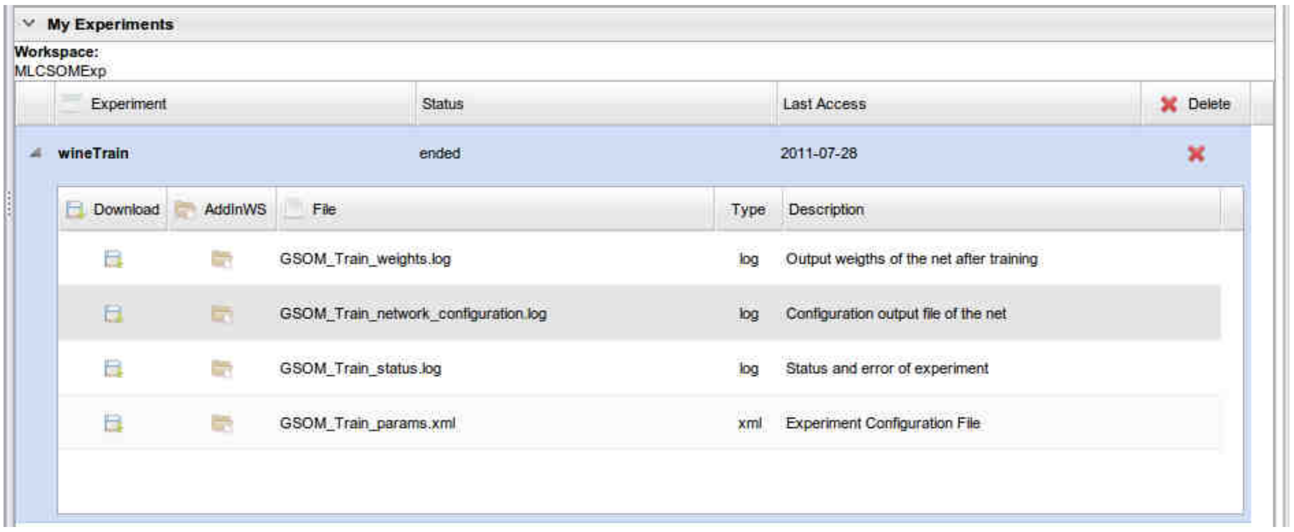
Fig. 8 – The wineTrain experiment configuration tab

Now we have to configure parameters for experiment. In particular, we will leave empty the not required fields (label without asterisk). The meaning of the parameters for this use case are described on section 3.6.1 of this document.
As alternative, you can click on the Help button to obtain detailed parameter description and their default value directly from the web application.

We give **columnSubset_wine.csv** as input dataset specifying 3 as number of clusters, because of the number of cultivars.

Once you have entered any required field, you can click on **submit** button and wait to the end of experiment.

The content of output files, obtained at the end of experiment (available when the experiment status is "ended") is shown in the following images. The meaning of the files is described on section 3.3.1.2


My Experiments
Workspace:
MLCSOMExp

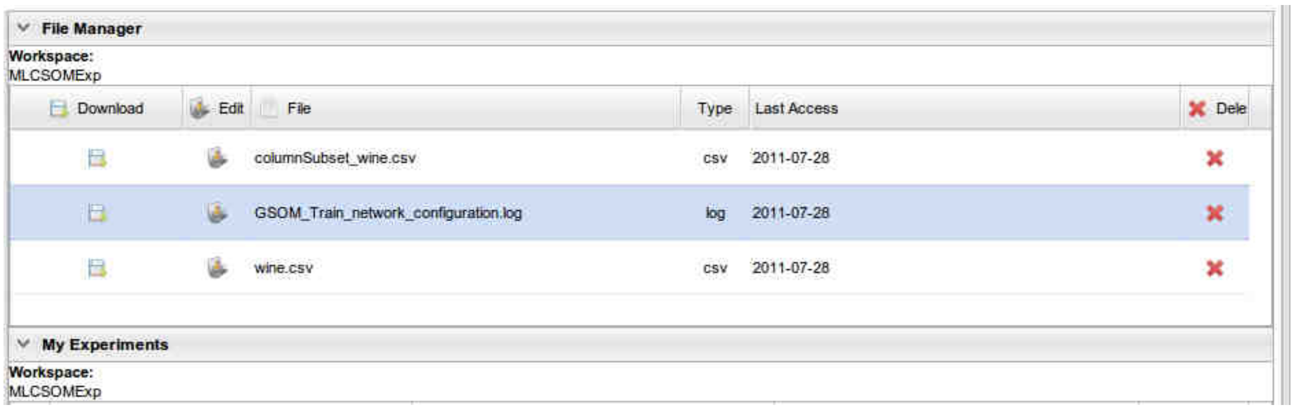
Experiment	Status	Last Access	Delete
wineTrain	ended	2011-07-28	✖

Download	AddInWS	File	Type	Description
		GSOM_Train_weights.log	log	Output weigths of the net after training
		GSOM_Train_network_configuration.log	log	Configuration output file of the net
		GSOM_Train_status.log	log	Status and error of experiment
		GSOM_Train_params.xml	xml	Experiment Configuration File

Fig. 9 – The wineTrain experiment output files

4.1.2 Clustering_GSOM Run Use Case

To execute a run experiment, you must put on input file area (**File Manager**) of the workspace the configuration file obtained as result of a training experiment. So, in this case we add the file named **GSOM_Train_Network_configuration.log**. This because it represent the stored configuration of the network, trained on input dataset file.

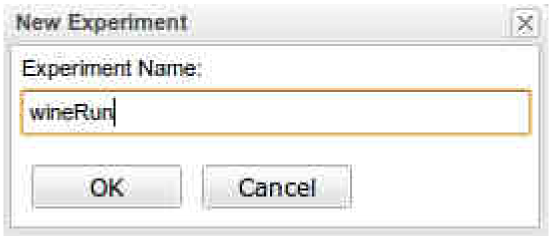

File Manager
Workspace:
MLCSOMExp

Download	Edit	File	Type	Last Access	Dele
		columnSubset_wine.csv	csv	2011-07-28	✖
		GSOM_Train_network_configuration.log	log	2011-07-28	✖
		wine.csv	csv	2011-07-28	✖

My Experiments
Workspace:
MLCSOMExp

Fig. 10 - The file GSOM_Train_network_configuration.log copied in the WS File Manager for Run experiment

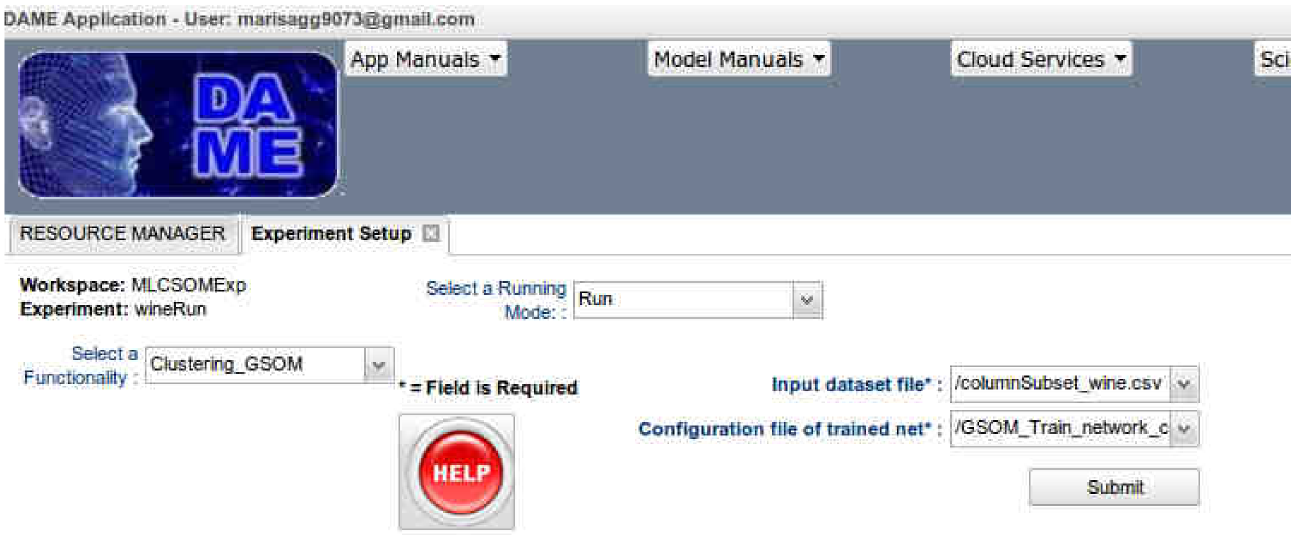
So far, we proceed to create a new experiment, named **wineRun**, to execute clustering.



Experiment Name: wineRun

Fig. 11 – The wineRun experiment creation tab

We will re-use the same input dataset (**column_subset_wine.csv**) and also select configuration file (**GSOM_Train_Network_configuration.log**).



Workspace: MLCSOMExp
Experiment: wineRun

Select a Running Mode: : Run

Select a Functionality : Clustering_GSOM

* = Field is Required

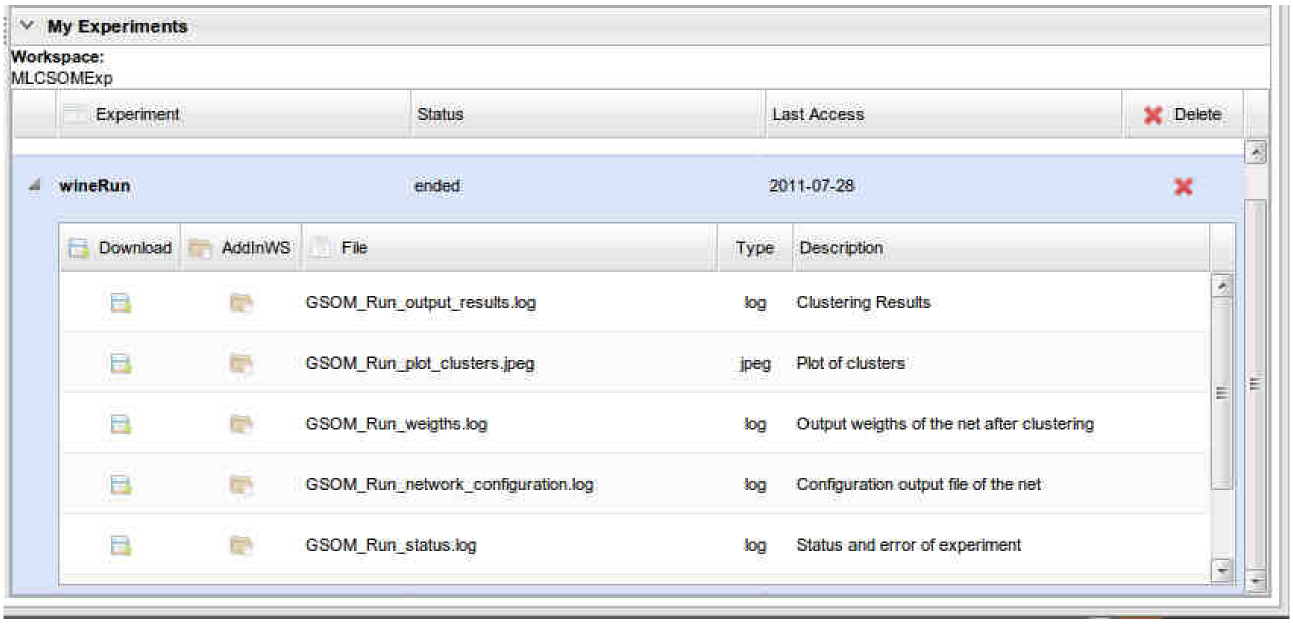
Input dataset file* : /columnSubset_wine.csv

Configuration file of trained net* : /GSOM_Train_network_c

Fig. 12 – The wineRun experiment configuration tab

Once you have entered any required field, you can click on **submit** button and wait to the end of experiment.

The content of output files, obtained at the end of experiment (available when the experiment status is "ended") is shown in the following images. The meaning of the files is described on section 3.3.2.2



My Experiments

Workspace: MLCSOMExp

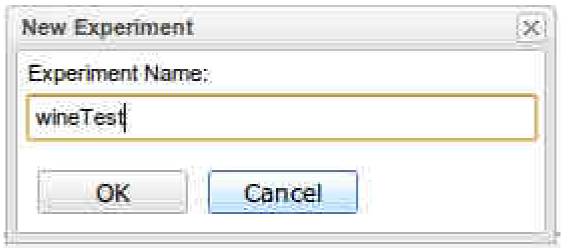
Experiment	Status	Last Access	Delete
wineRun	ended	2011-07-28	✖

Download	AddInWS	File	Type	Description
		GSOM_Run_output_results.log	log	Clustering Results
		GSOM_Run_plot_clusters.jpeg	jpeg	Plot of clusters
		GSOM_Run_weigths.log	log	Output weigths of the net after clustering
		GSOM_Run_network_configuration.log	log	Configuration output file of the net
		GSOM_Run_status.log	log	Status and error of experiment

Fig. 13 – The wineRun experiment output files

4.1.3 Clustering_GSOM Test Use Case

To execute a test experiment, you must put on input file area (File Manager) of the workspace the configuration file obtained as result of a training experiment. So, in this case we use the file **GSOM_Train_Network_configuration.log,** already used on run use case.
So far, we proceed to create a new experiment, named **wineTest**, to execute test use case.

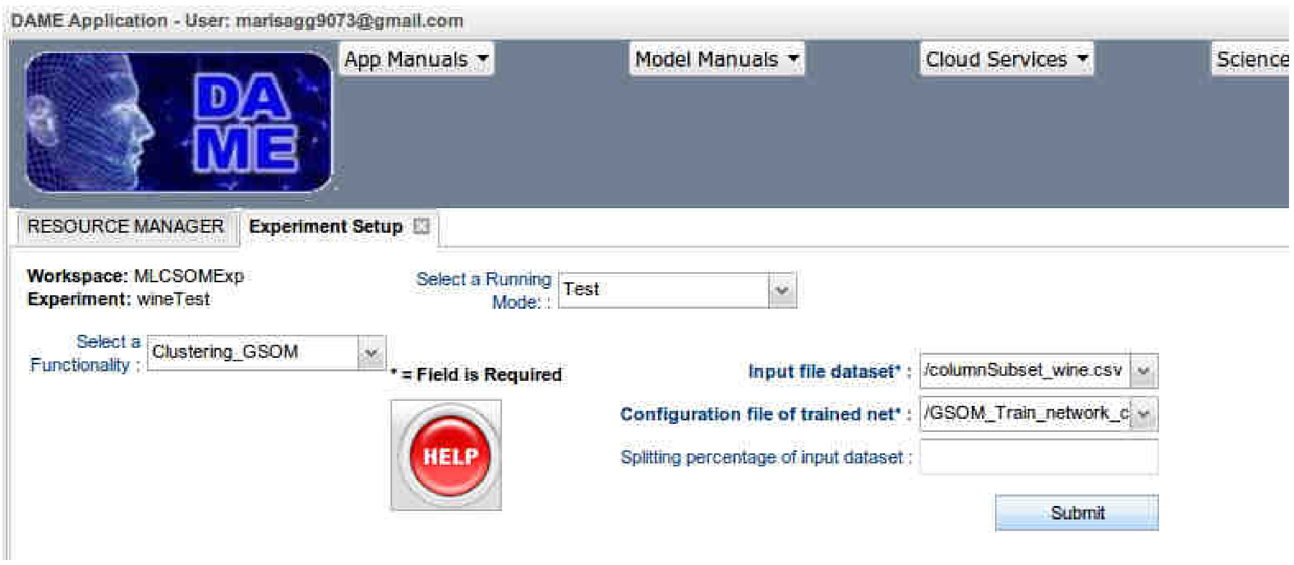


New Experiment

Experiment Name: wineTest

Fig. 14 – The wineTest experiment creation tab

We will re-use the same input dataset (**column_subset_wine.csv**) and also select configuration file (**GSOM_Train_Network_configuration.log**).



DAME Application - User: marisagg9073@gmail.com

Workspace: MLCSOMExp
Experiment: wineTest

Select a Running Mode: : Test

Select a Functionality : Clustering_GSOM

* = Field is Required

Input file dataset* : /columnSubset_wine.csv

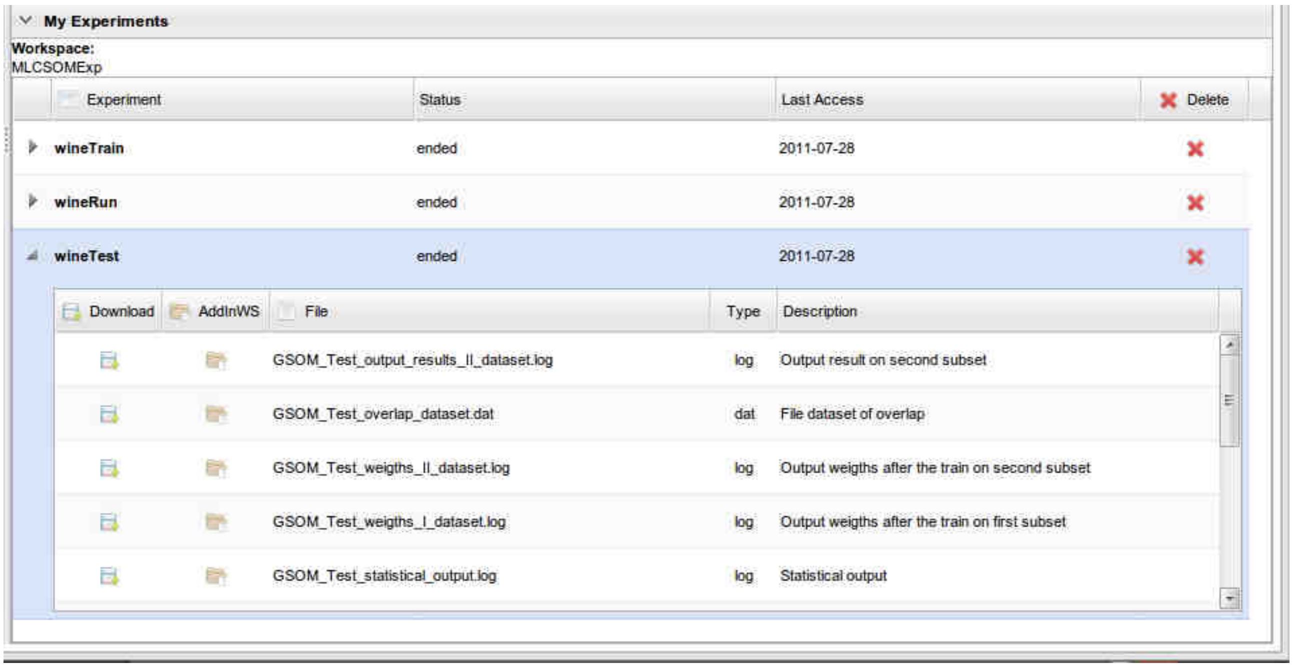
Configuration file of trained net* : /GSOM_Train_network_c

Splitting percentage of input dataset :

Fig. 15 – The wineTest experiment configuration tab

Once you have entered any required field, you can click on **submit** button and wait to the end of experiment.

The content of output files, obtained at the end of experiment (available when the experiment status is "ended") is shown in the following images. The meaning of the files is described on section 3.3.3.2.



My Experiments

Workspace:
MLCSOMExp

Experiment	Status	Last Access	Delete
wineTrain	ended	2011-07-28	✖
wineRun	ended	2011-07-28	✖
wineTest	ended	2011-07-28	✖

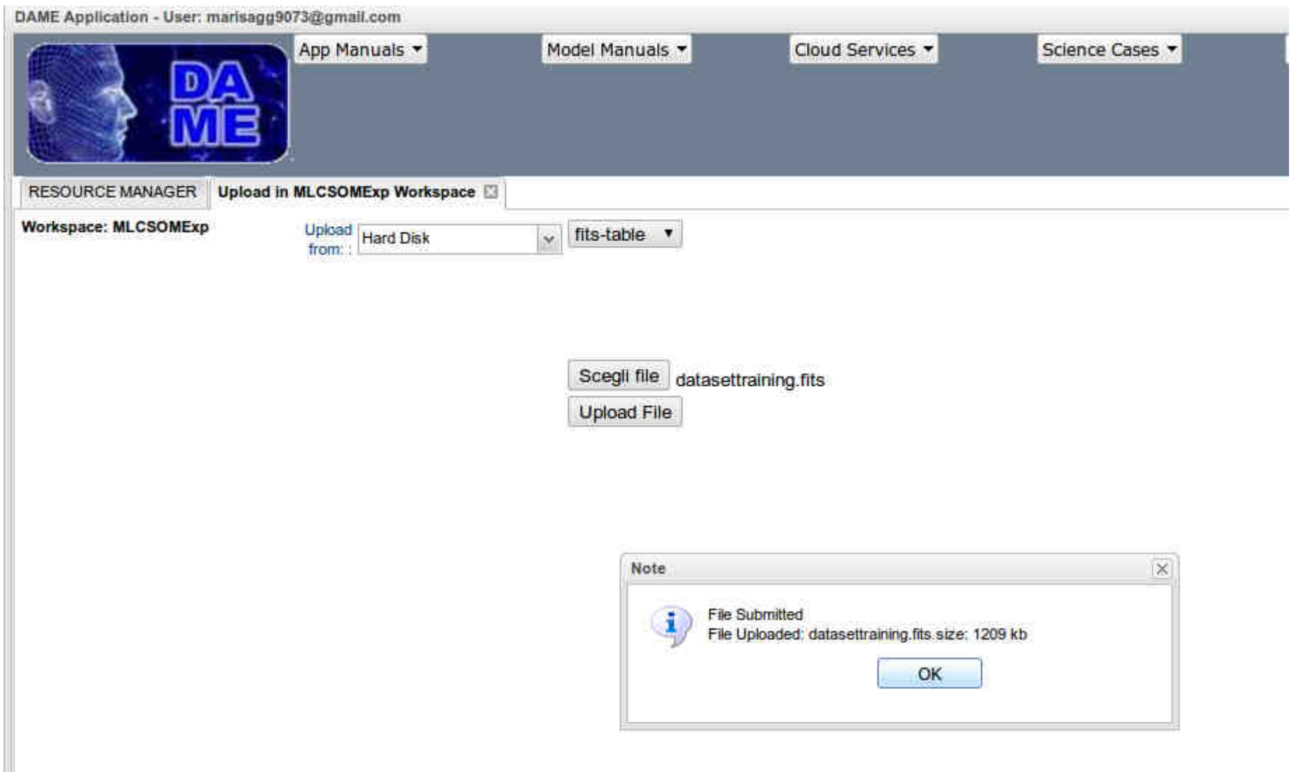
Download	AddInWS	File	Type	Description
		GSOM_Test_output_results_II_dataset.log	log	Output result on second subset
		GSOM_Test_overlap_dataset.dat	dat	File dataset of overlap
		GSOM_Test_weigths_II_dataset.log	log	Output weigths after the train on second subset
		GSOM_Test_weigths_I_dataset.log	log	Output weigths after the train on first subset
		GSOM_Test_statistical_output.log	log	Statistical output

Fig. 16 – The wineTest experiment output files

4.2 Usage Example of Clustering_CSOM

The model CSOM is specialised to work on FITS files (Flexible …). So, we will show an example of experiment on dataset_training.fits, available for upload on the website http://dame.dsf.unina.it/beta_info.html , under the heading "**Test Resources**".

First of all, you must upload the file on **MLCSOMExp** workspace.



datasettraining.fits

File Submitted
File Uploaded: datasettraining.fits size: 1209 kb

Fig. 17 – Upload Manager: uploading datasettraining.fits file

4.2.1 Clustering_CSOM Train Use Case

Once uploaded input file dataset on workspace, you can create a new experiment, named **fitsTrain**.

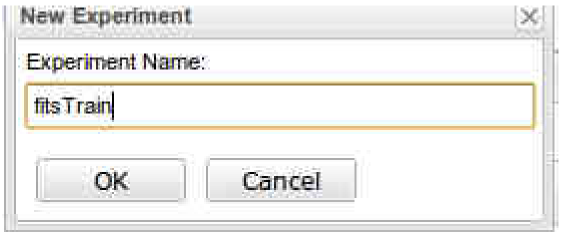


Fig. 18 – The fitsTrain experiment creation tab

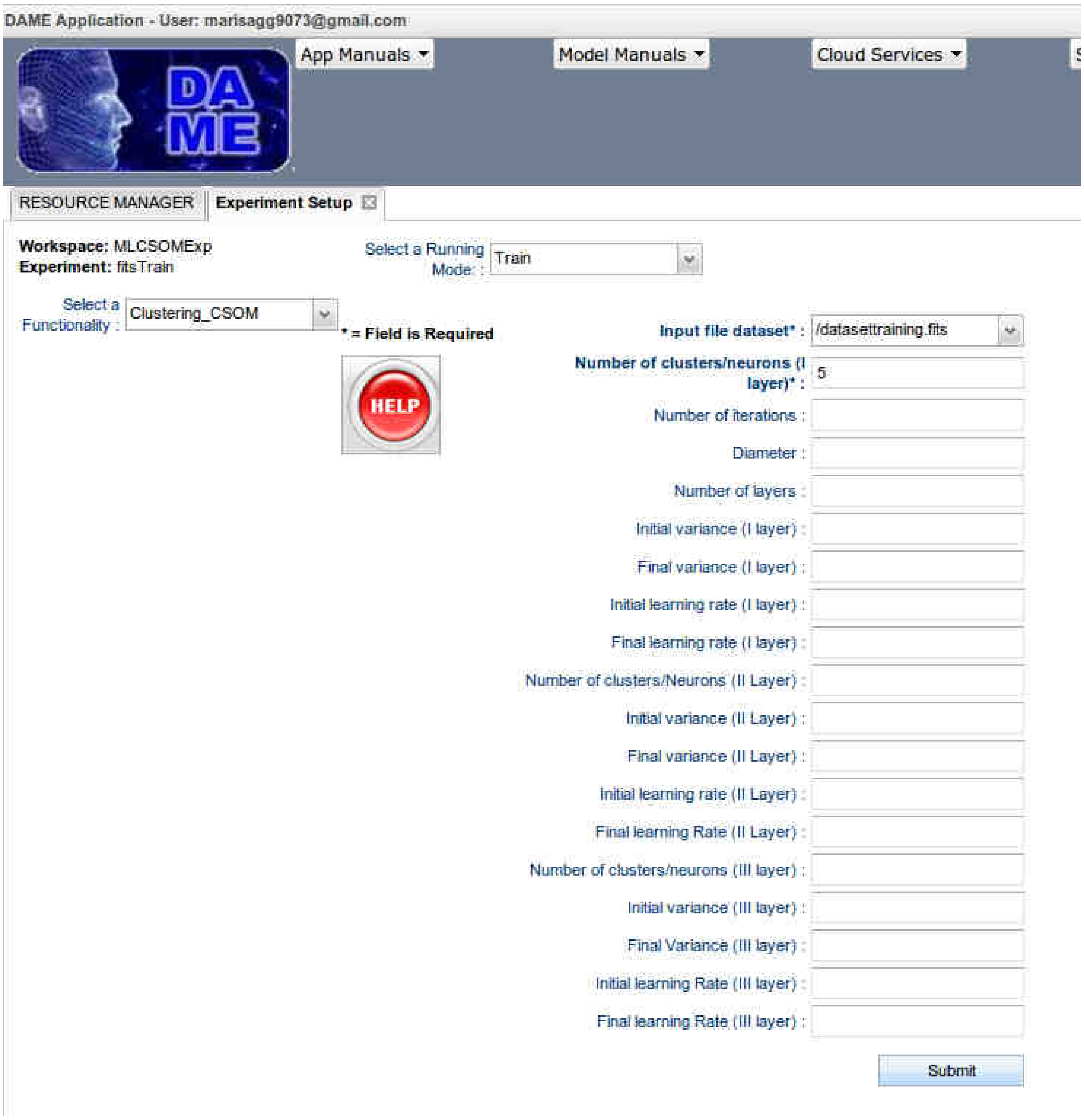
After creation, the new configuration tab is open. Here we select Clustering_CSOM as couple functionality-model of the current experiment and we select also Train use case.

Now we have to configure parameters for experiment. In particular, we will leave empty the not required fields (label without asterisk). The meaning of the parameters for this use case are described on section 3.6.1 of this document.
As alternative, you can click on the Help button to obtain detailed parameter description and their default value directly from the web application.

We give **dataset_training.fits** as input dataset specifying 5 as number of clusters (see).

Once you have entered any required field, you can click on **submit** button and wait to the end of experiment.

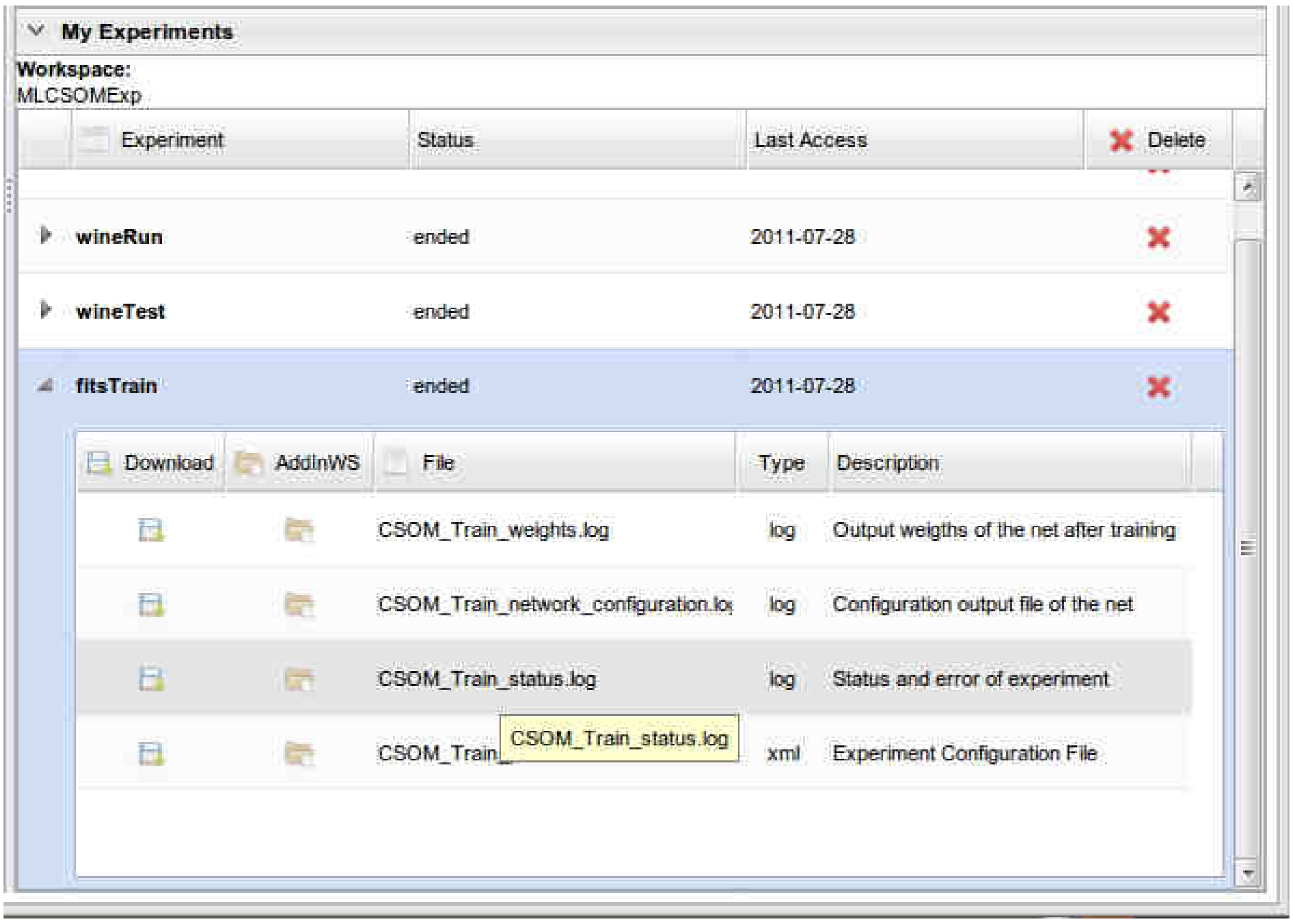
The content of output files, obtained at the end of experiment (available when the experiment status is "ended") is shown in (). The meaning of the files is described on section 3.4.1.2.


DAME Application - User: marisagg9073@gmail.com

Workspace: MLCSOMExp
Experiment: fitsTrain

Select a Running Mode:	Train
Select a Functionality:	Clustering_CSOM
* = Field is Required	
Input file dataset* :	/datasettraining.fits
Number of clusters/neurons (I layer)* :	5
Number of iterations :	
Diameter :	
Number of layers :	
Initial variance (I layer) :	
Final variance (I layer) :	
Initial learning rate (I layer) :	
Final learning rate (I layer) :	
Number of clusters/Neurons (II Layer) :	
Initial variance (II Layer) :	
Final variance (II Layer) :	
Initial learning rate (II Layer) :	
Final learning Rate (II Layer) :	
Number of clusters/neurons (III layer) :	
Initial variance (III layer) :	
Final Variance (III layer) :	
Initial learning Rate (III layer) :	
Final learning Rate (III layer) :	

Fig. 19 – The fitsTrain experiment configuration tab



My Experiments

Workspace: MLCSOMExp

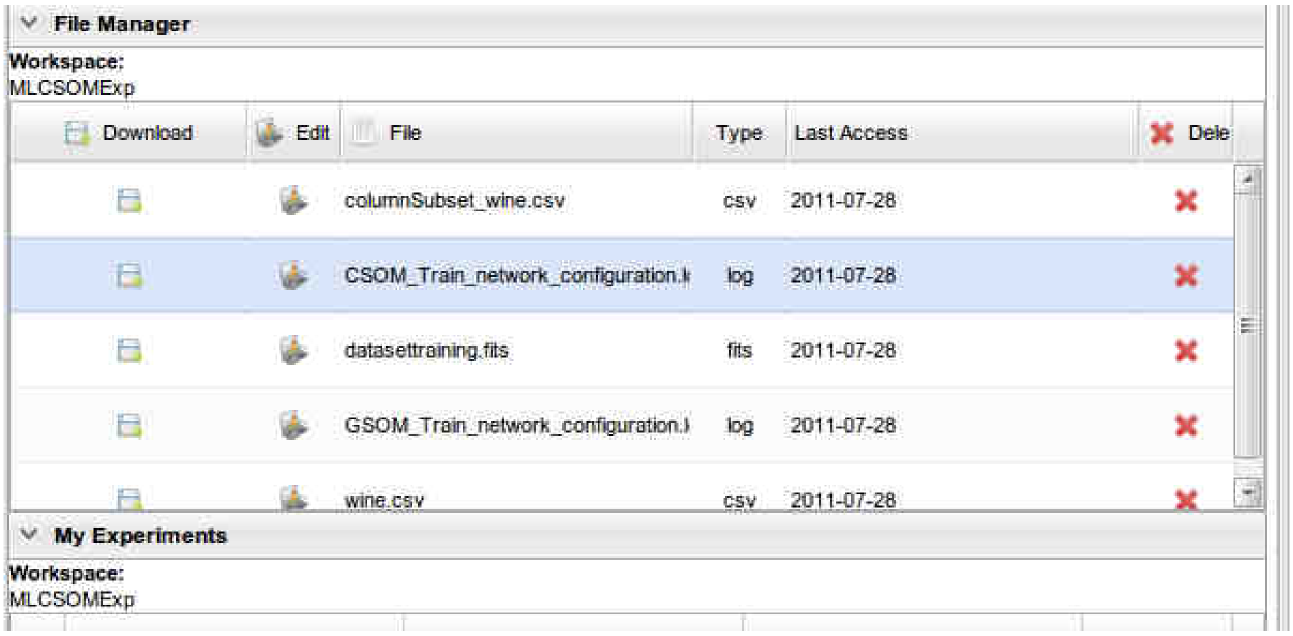
Experiment	Status	Last Access	Delete
wineRun	ended	2011-07-28	✖
wineTest	ended	2011-07-28	✖
fitsTrain	ended	2011-07-28	✖

Download	AddInWS	File	Type	Description
		CSOM_Train_weights.log	log	Output weigths of the net after training
		CSOM_Train_network_configuration.lo	log	Configuration output file of the net
		CSOM_Train_status.log	log	Status and error of experiment
		CSOM_Train_	xml	Experiment Configuration File

Fig. 20 – The fitsTrain experiment output files

4.2.2 Clustering_CSOM Run Use Case

To excute a run experiment,you must put on input file area (**File Manager**) of the workspace the configuration file obtained as result of a training experiment. So, in this case we add the file named **CSOM_Train_Network_configuration.log**. This because it represent the stored configuration of the network, trained on input dataset file.



File Manager

Workspace:
MLCSOMExp

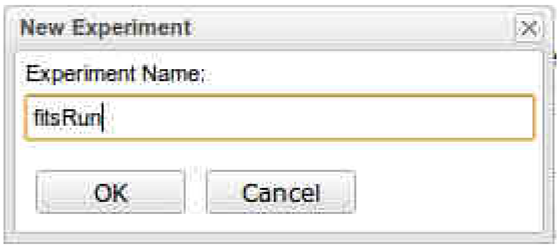
Download	Edit	File	Type	Last Access	Dele
		columnSubset_wine.csv	csv	2011-07-28	
		CSOM_Train_network_configuration.l	log	2011-07-28	
		datasettraining.fits	fits	2011-07-28	
		GSOM_Train_network_configuration.l	log	2011-07-28	
		wine.csv	csv	2011-07-28	

My Experiments

Workspace:
MLCSOMExp

Fig. 21 – File Manager: after uploading of CSOM_Train_Network_configuration.log

So far, we proceed to create a new experiment, named **fitsRun**, to execute clustering.



New Experiment

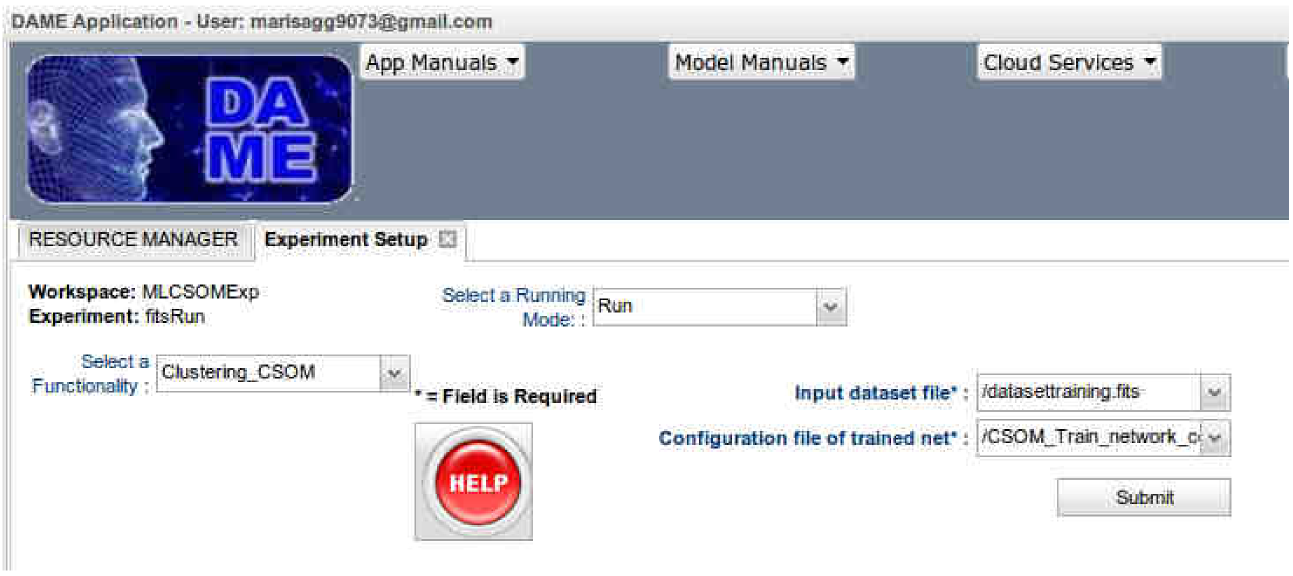
Experiment Name:
fitsRun

Fig. 22 – The fitsRun experiment creation tab

We will re-use the same input dataset (**datasettraining.fits**) and also select configuration file (**CSOM_Train_Network_configuration.log**). ()

Once you have entered any required field, you can click on **submit** button and wait to the end of experiment.

The content of output files, obtained at the end of experiment (available when the experiment status is "ended") is shown in the following images. The meaning of the files is described on section 3.4.2.2.



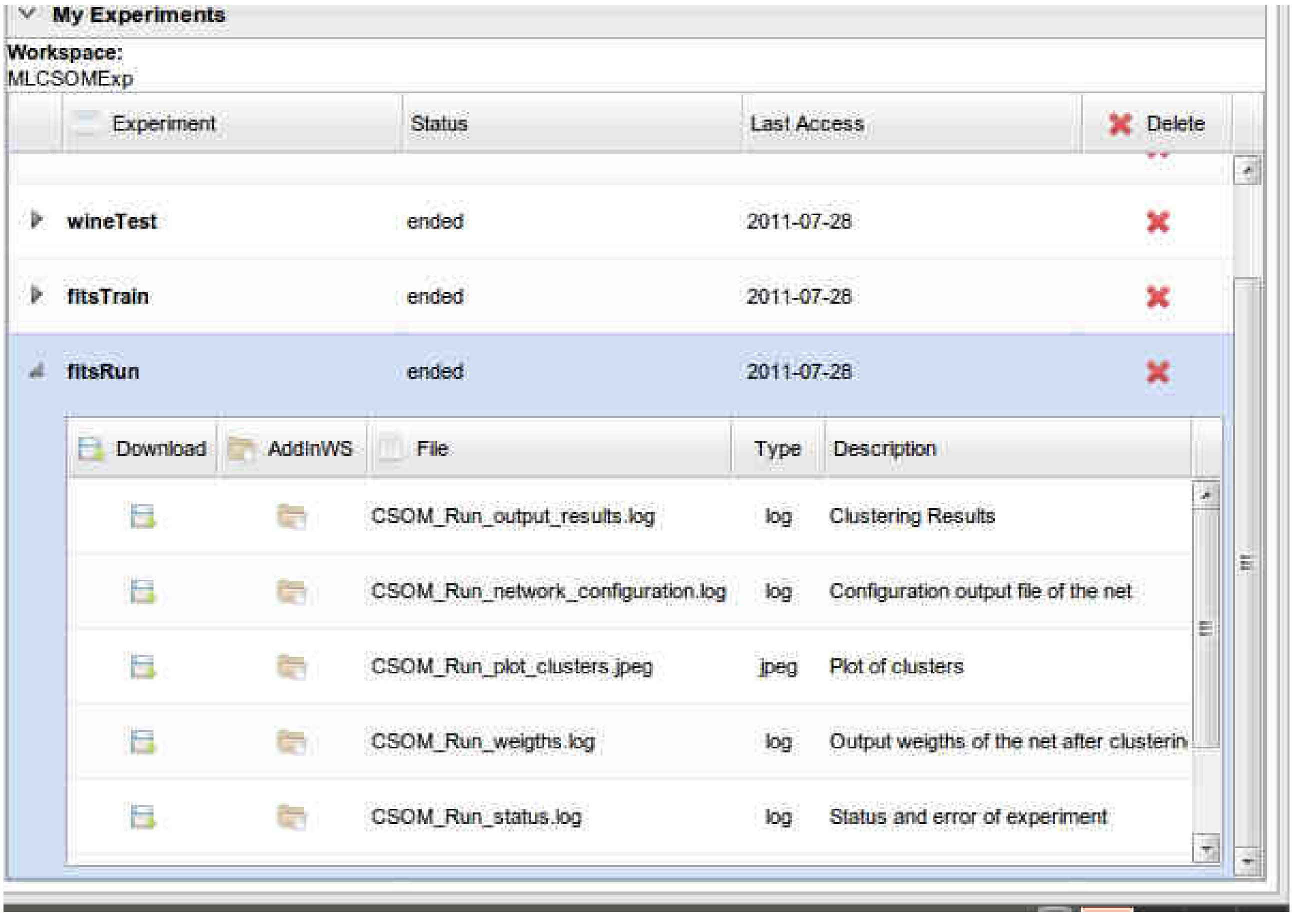
DAME Application - User: marisagg9073@gmail.com

RESOURCE MANAGER | Experiment Setup

Workspace: MLCSOMExp
Experiment: fitsRun

Select a Running Mode:	Run
Select a Functionality :	Clustering_CSOM
Input dataset file* :	/datasettraining.fits
Configuration file of trained net* :	/CSOM_Train_network_c

Fig. 23 – The fitsRun experiment configuration tab



My Experiments
Workspace:
MLCSOMExp

Experiment	Status	Last Access	Delete
wineTest	ended	2011-07-28	✖
fitsTrain	ended	2011-07-28	✖
fitsRun	ended	2011-07-28	✖

Download	AddInWS	File	Type	Description
		CSOM_Run_output_results.log	log	Clustering Results
		CSOM_Run_network_configuration.log	log	Configuration output file of the net
		CSOM_Run_plot_clusters.jpeg	jpeg	Plot of clusters
		CSOM_Run_weigths.log	log	Output weigths of the net after clusterin
		CSOM_Run_status.log	log	Status and error of experiment

Fig. 24 – The fitsRun experiment output files

4.2.3 Clustering_CSOM Test Use Case

To execute a test experiment, you must put on input file area (**File Manager**) of the workspace the configuration file obtained as result of a training experiment. So, in this case we use the file **CSOM_Train_Network_configuration.log,** already used on run use case.
So far, we proceed to create a new experiment, named **fitsTest**, to execute test use case.

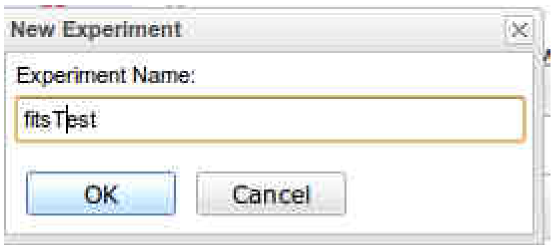
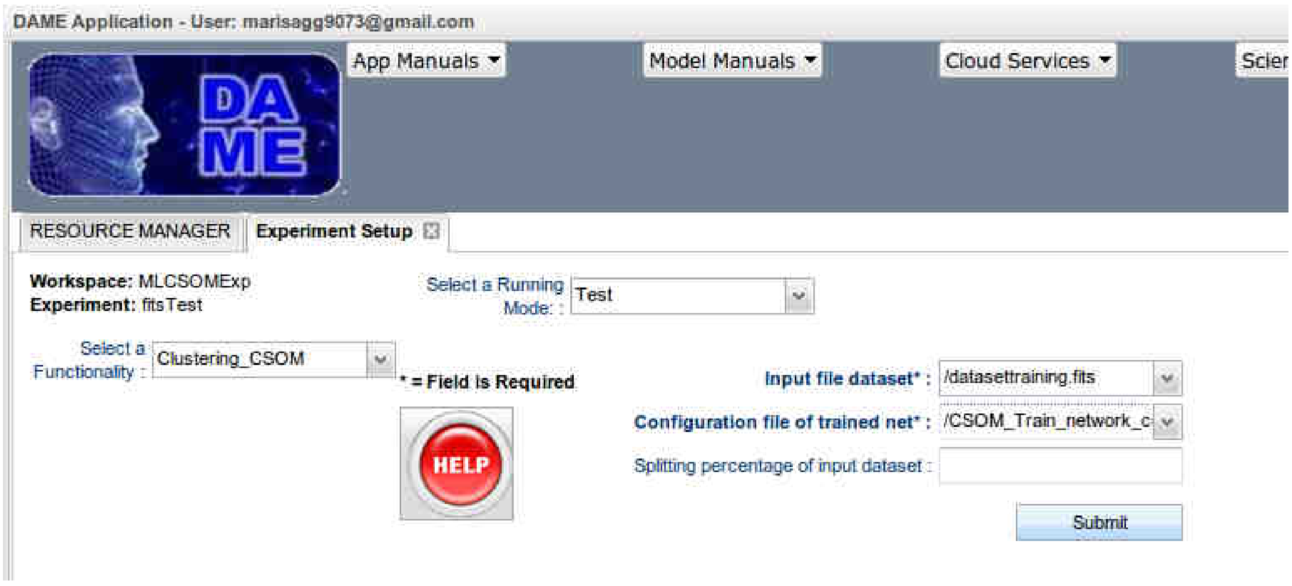


Fig. 25 – The fitsTest experiment creation tab

We will re-use the same input dataset (**datasettraining.fits**) and also select configuration file (**CSOM_Train_Network_configuration.log**).

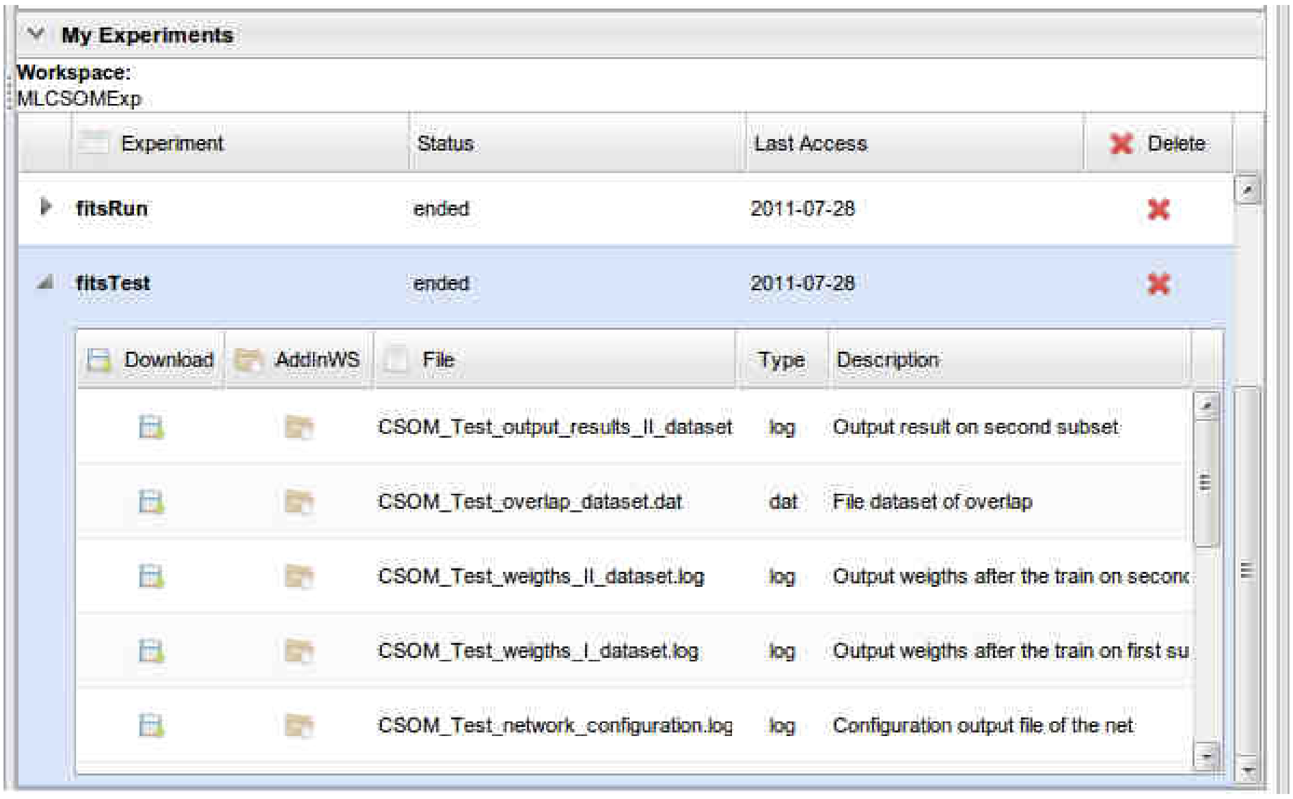

DAME Application - User: marisagg9073@gmail.com

Workspace: MLCSOMExp
Experiment: fitsTest
Select a Running Mode: Test
Select a Functionality: Clustering_CSOM
* = Field is Required
Input file dataset* : /datasettraining.fits
Configuration file of trained net* : /CSOM_Train_network_c
Splitting percentage of input dataset :

Fig. 26 – The fitsTest experiment configuration tab

Once you have entered any required field, you can click on **submit** button and wait to the end of experiment.

The content of output files, obtained at the end of experiment (available when the experiment status is "ended") is shown in the following images. The meaning of the files is described on section 3.4.3.23.3.3.2



Experiment	Status	Last Access	Delete
fitsRun	ended	2011-07-28	✖
fitsTest	ended	2011-07-28	✖

Download	AddInWS	File	Type	Description
		CSOM_Test_output_results_II_dataset	log	Output result on second subset
		CSOM_Test_overlap_dataset.dat	dat	File dataset of overlap
		CSOM_Test_weigths_II_dataset.log	log	Output weigths after the train on seconc
		CSOM_Test_weigths_I_dataset.log	log	Output weigths after the train on first su
		CSOM_Test_network_configuration.log	log	Configuration output file of the net

Fig. 27 – The fitsTest experiment output files

5 Appendix – References and Acronyms

Abbreviations & Acronyms

A & A	Meaning	A & A	Meaning
AI	Artificial Intelligence	KDD	Knowledge Discovery in Databases
ANN	Artificial Neural Network	IEEE	Institute of Electrical and Electronic Engineers
ARFF	Attribute Relation File Format	INAF	Istituto Nazionale di Astrofisica
ASCII	American Standard Code for Information Interchange	JPEG	Joint Photographic Experts Group
BoK	Base of Knowledge	LAR	Layered Application Architecture
BP	Back Propagation	MDS	Massive Data Sets
BLL	Business Logic Layer	MLC	Multi Layer Clustering
CE	Cross Entropy	MLP	Multi Layer Perceptron
CSOM	Clustering SOM	MSE	Mean Square Error
CSV	Comma Separated Values	NN	Neural Network
DAL	Data Access Layer	OAC	Osservatorio Astronomico di Capodimonte
DAME	DAta Mining & Exploration	PC	Personal Computer
DAMEWARE	DAME Web Application REsource	PI	Principal Investigator
DAPL	Data Access & Process Layer	REDB	Registry & Database
DL	Data Layer	RIA	Rich Internet Application
DM	Data Mining	SDSS	Sloan Digital Sky Survey
DMM	Data Mining Model	SL	Service Layer
DMS	Data Mining Suite	SOFM	Self Organizing Feature Map
FITS	Flexible Image Transport System	SOM	Self Organizing Map
FL	Frontend Layer	SW	Software
FW	FrameWork	UI	User Interface
GRID	Global Resource Information Database	URI	Uniform Resource Indicator
GSOM	Gated SOM	VO	Virtual Observatory
GUI	Graphical User Interface	XML	eXtensible Markup Language
HW	Hardware		

Tab. 3 – Abbreviations and acronyms

Reference & Applicable Documents

ID	Title / Code	Author	Date
R1	"The Use of Multiple Measurements in Taxonomic Problems", in Annals of Eugenics, 7, p. 179--188	Ronald Fisher	1936
R2	*Neural Networks for Pattern Recognition*. Oxford University Press, GB	Bishop, C. M.	1995
R3	*Neural Computation*	Bishop, C. M., Svensen, M. & Williams, C. K. I.	1998
R4	Data Mining Introductory and Advanced Topics, Prentice-Hall	Dunham, M.	2002
R5	Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:-69	T. Kohonen	1982
R6	How patterned neural connections can be set up by self-organization. In *Proceedings of the Royal Society London*, volume B194, pages 431-445	D. J. Willshaw and C. von der Malsburg	1976
R7	Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):-1460	B. Fritzke	1994
R8	*The Fourth Paradigm*. Microsoft research, Redmond Washington, USA	Hey, T. et al.	2009
R9	Artificial Intelligence, A modern Approach. Second ed. (Prentice Hall)	Russell, S., Norvig, P.	2003
R10	Neural Networks - A comprehensive Foundation, Second Edition, Prentice Hall	Haykin, S.,	1999
R11	*A practical application of simulated annealing to clustering*. Pattern Recognition 25(4): 401-412	Donald E. Brown D.E., Huntley, C. L.:	1991

Tab. 4 – Reference Documents

ID	Title / Code	Author	Date
A1	SuiteDesign_VONEURAL-PDD-NA-0001-Rel2.0	DAME Working Group	15/10/2008
A2	project_plan_VONEURAL-PLA-NA-0001-Rel2.0	Brescia	19/02/2008
A3	statement_of_work_VONEURAL-SOW-NA-0001-Rel1.0	Brescia	30/05/2007
A4	MLP_user_manual_VONEURAL-MAN-NA-0001-Rel1.0	DAME Working Group	12/10/2007
A5	pipeline_test_VONEURAL-PRO-NA-0001-Rel.1.0	D'Abrusco	17/07/2007
A6	scientific_example_VONEURAL-PRO-NA-0002-Rel.1.1	D'Abrusco/Cavuoti	06/10/2007
A7	frontend_VONEURAL-SDD-NA-0004-Rel1.4	Manna	18/03/2009
A8	FW_VONEURAL-SDD-NA-0005-Rel2.0	Fiore	14/04/2010
A9	REDB_VONEURAL-SDD-NA-0006-Rel1.5	Nocella	29/03/2010
A10	driver_VONEURAL-SDD-NA-0007-Rel0.6	d'Angelo	03/06/2009
A11	dm-model_VONEURAL-SDD-NA-0008-Rel2.0	Cavuoti/Di Guido	22/03/2010
A12	ConfusionMatrixLib_VONEURAL-SPE-NA-0001-Rel1.0	Cavuoti	07/07/2007
A13	softmax_entropy_VONEURAL-SPE-NA-0004-Rel1.0	Skordovski	02/10/2007
A14	VONeuralMLP2.0_VONEURAL-SPE-NA-0007-Rel1.0	Skordovski	20/02/2008
A15	dm_model_VONEURAL-SRS-NA-0005-Rel0.4	Cavuoti	05/01/2009
A16	FANN_MLP_VONEURAL-TRE-NA-0011-Rel1.0	Skordovski, Laurino	30/11/2008
A17	DMPlugins_DAME-TRE-NA-0016-Rel0.3	Di Guido, Brescia	14/04/2010
A18	BetaRelease_ReferenceGuide_DAME-MAN-NA-0009-Rel1.0	Brescia	28/10/2010
A19	BetaRelease_GUI_UserManual_DAME-MAN-NA-0010-Rel1.0	Brescia	03/12/2010
A20	MLCSOM_VONEURAL-SDD-NA-0017-Rel1.0	Guglielmo	29/06/2011

Tab. 5 – Applicable Documents

__oOo__

DAta Mining & Exploration Program

DAME Program
“we make science discovery happen”