DAta Mining & Exploration Program

Dipartimento di Scienze Fisiche
Università di Napoli "Federico II"

ISTITUTO NAZIONALE di ASTROFISICA
OSSERVATORIO ASTRONOMICO di CAPODIMONTE

CALTECH

Support Vector Machines

User Manual

DAME-MAN-NA-0012

Issue: 1.4
Date: September 23, 2013
Author: S. Cavuoti

Doc. : SVM_UserManual_DAME-MAN-NA-0013-Rel1.4

INDEX

TABLE INDEX

1 Introduction

The present document is the user guide of the data mining model SVM (Support Vector Machines), as implemented by LibSVM [A13] and integrated into the DAMEWARE.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 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 SVM Model Theoretical Overview

This paragraph is intended to furnish a theoretical overview of the SVM model, associated to single or multiple functionality domains, in order to be used to perform practical scientific experiments with such techniques. An overview of machine learning and functionality domains, as intended in DAME Project can be found in [A18].

2.1 SVM Classification

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. In a short period of time, SVM found numerous applications in a lot of scientific branches like physics, biology, chemistry.

- drug design (discriminating between ligands and nonligands, inhibitors and noninhibitors, etc.),
- quantitative structure-activity relationships (QSAR, where SVM regression is used to predict various physical, chemical, or biological properties),
- chemometrics (optimization of chromatographic separation or compound concentration prediction from spectral data as examples),
- sensors (for qualitative and quantitative prediction from sensor data),
- chemical engineering (fault detection and modeling of industrial processes),
- text mining (automatic recognition of scientific information)
- etc.

SVM models were originally defined for the classification of linearly separable classes of objects. For any particular set of two-class objects, an SVM finds the unique hyperplane having the maximum margin. H3 (green) doesn't separate the 2 classes. H1 (blue) does, with a small margin and H2 (red) with the maximum margin.

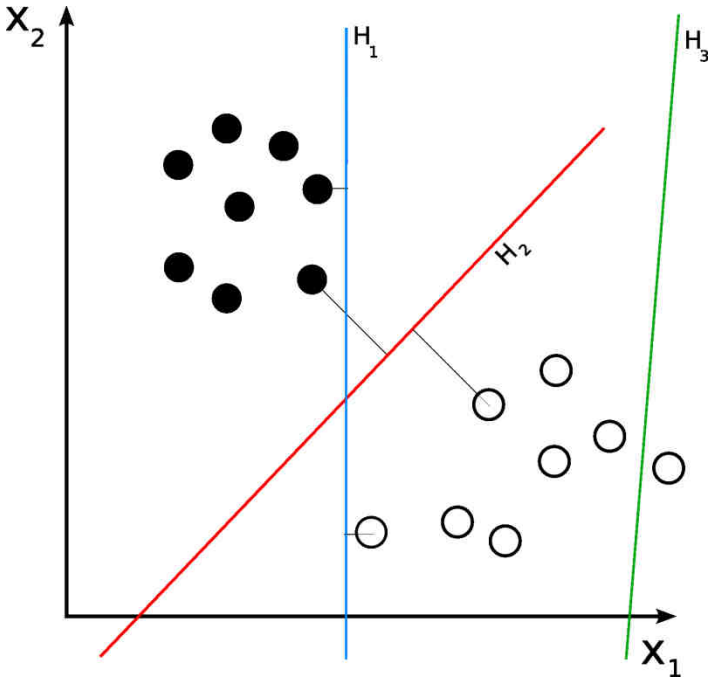


Fig. 1 H3 (green) doesn't separate the 2 classes. H1 (blue) with a small margin and H2 (red) with the maximum margin.

The hyperplane H1 defines the border with class +1 objects, whereas the hyperplane H2 defines the border with class 1 objects. Two objects from class +1 define the hyperplane H1, and three objects from class -1 define the hyperplane H2. These objects, represented inside circles in Figure, are called support vectors. A special characteristic of SVM is that the solution to a classification problem is represented by the support vectors that determine the maximum margin hyperplane.

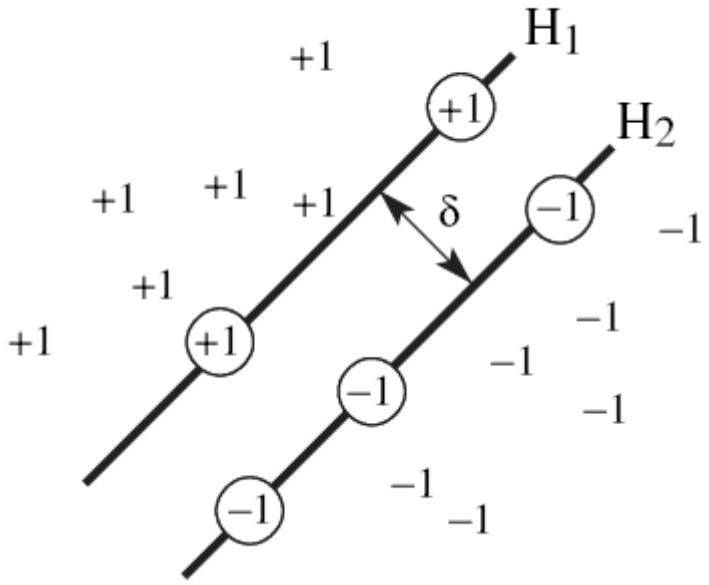


Fig. 2 representation of support vectors

In a plane, combinations of three points from two classes can be separated with a line. Four points cannot be separated with a linear classifier.

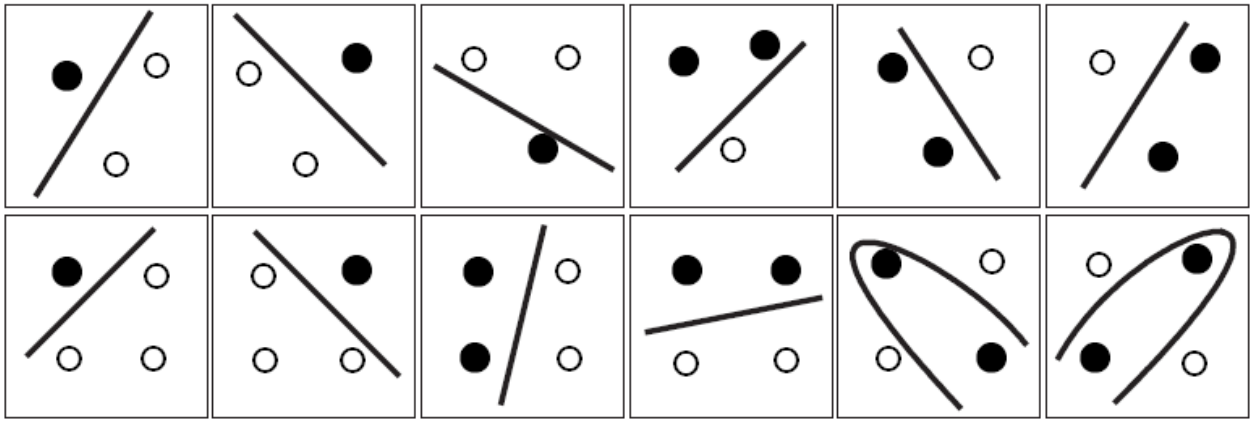


Fig. 3 Not all the points combinations may be separated by a linear classifier

SVM can also be used to separate classes that cannot be separated with a linear classifier. In such cases, the coordinates of the objects are mapped into a feature space using nonlinear functions called feature functions ϕ. The feature space is a high-dimensional space in which the two classes can be separated with a linear classifier.

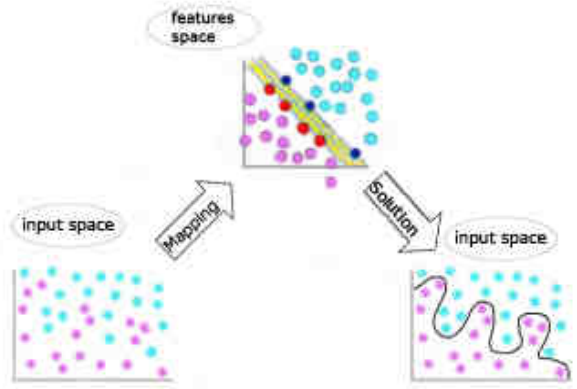


Fig. 4 how the feature function works

The nonlinear feature function ϕ combines the input space (the original coordinates of the objects) into the feature space, which can even have an infinite dimension. Because the feature space is high dimensional, it is not practical to use directly feature functions ϕ in computing the classification hyperplane. Instead, the nonlinear mapping induced by the feature functions is computed with special functions called kernels. Kernels have the advantage of operating in the input space, where the solution of the classification problem is a weighted sum of kernel functions evaluated at the support vectors.

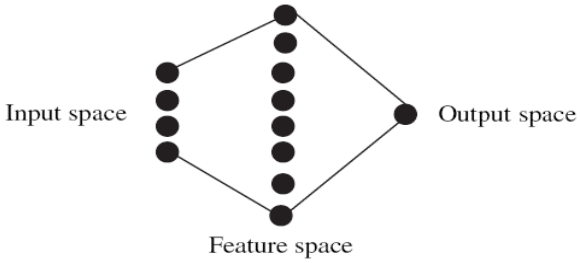


Fig. 5 relation between input and output space (by the mediation of the feature space)

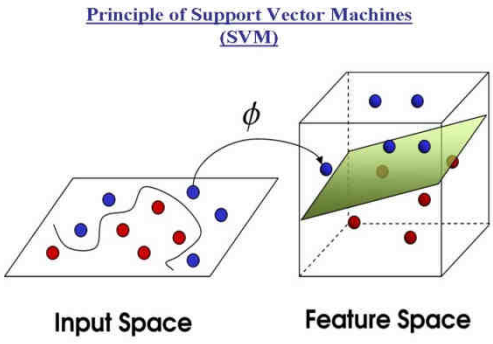


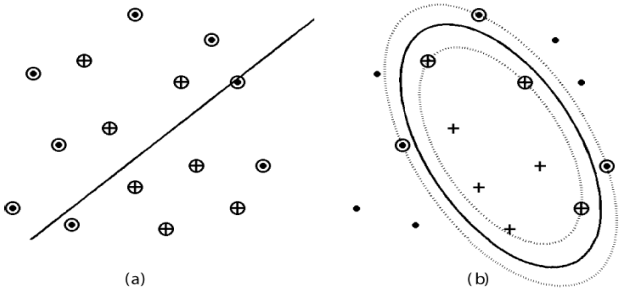
Fig. 6 another representation of the mapping work

The LibSVM implementation of SVM we use has 4 kernels:

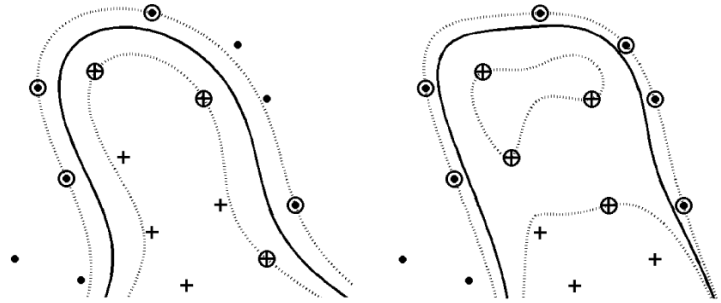
- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)^d$, $\gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)$.

To illustrate the SVM capability of training nonlinear classifiers, consider the patterns from Table. This is a synthetic dataset of two-dimensional patterns, designed to investigate the properties of the SVM classification method. In all figures, class +1 patterns are represented by + , whereas class -1 patterns are represented by black dots. The SVM hyperplane is drawn with a continuous line, whereas the margins of the SVM hyperplane are represented by dotted lines. Support vectors from the class +1 are represented as + inside a circle, whereas support vectors from the class -1 are represented as a black dot inside a circle

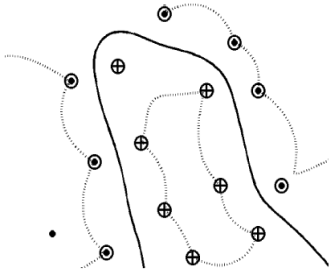
Pattern	x_1	x_2	Class
1	2	4.5	1
2	2.5	2.9	1
3	3	1.5	1
4	3.6	0.5	1
5	4.2	2	1
6	3.9	4	1
7	5	1	1
8	0.6	1	−1
9	1	4.2	−1
10	1.5	2.5	−1
11	1.75	0.6	−1
12	3	5.6	−1
13	4.5	5	−1
14	5	4	−1
15	5.5	2	−1



Linear Polynomial Degree = 2



Polynomial degree = 3 Polynomial degree = 10



Radial Basis Function Gamma = 0.5

As we can see the linear kernel doesn't works in this example, the other 4 tests discriminate perfectly the two classes but we can see that solutions are quite different each other, is important have a test set in order to choose the best one and avoid the over – fitting, the other bad news is that kernel functions (except the linear one) are not properly functions but family of functions so we need to try various parameters (usually called Hyper Parameter) to make the best choice

2.1.1 C-Support Vector Classification (C-SVC)

Given training vectors $\mathbf{x}_i \in R^n, i = 1, \ldots, l,$ in two classes, and a vector $\mathbf{y} \in R^l$ such that $y_i \in \{1, -1\}$, C-SVC (Boser et al., 1992; Cortes and Vapnik, 1995) solves the following primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l. \end{aligned}$$

It's dual is:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - \mathbf{e}^T\alpha \\ \text{subject to} \quad & \mathbf{y}^T\alpha = 0, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l, \end{aligned}$$

where e is the vector of all ones, C > 0 is the upper bound,

Q is an l by l positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$,

and $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is the kernel.

Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ. The decision function is:

$$\operatorname{sgn}\left(\sum_{i=1}^{l} y_i\alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

2.1.2 v-Support Vector Classification (v-SVC)

The ν-support vector classification (Scholkopf et al., 2000) uses a new parameter ν which controls the number of support vectors and training errors. The parameter ν ϵ (0; 1] is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.
Given training vectors $x_i \epsilon R^n$, i = 1,…,l, in two classes, and a vector y ϵ R^l such that $y_i \epsilon\{1, -1\}$ the primal form considered is:

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\rho} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq \rho - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l, \rho \geq 0. \end{aligned}$$

The dual is:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq 1/l, \qquad i = 1, \ldots, l, \\ & \mathbf{e}^T\alpha \geq \nu, \quad \mathbf{y}^T\alpha = 0. \\ & \text{where } Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

The decision function is:

$$\text{sgn}\left(\sum_{i=1}^{l} y_i\alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

2.1.3 Distribution Estimation (One-Class SVM)

One-class SVM was proposed by Scholkopf et al. (2001) for estimating the support of a high-dimensional distribution. Given training vectors $x_i \epsilon R^n$; i = 1,…, l without any class information, the primal form in (Scholkopf et al., 2001) is:

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\rho} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} - \rho + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & \mathbf{w}^T\phi(\mathbf{x}_i) \geq \rho - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l. \end{aligned}$$

The dual is:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq 1/(\nu l), i = 1, \ldots, l, \\ & \mathbf{e}^T\alpha = 1, \\ & \text{where } Q_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j). \end{aligned}$$

2.2 SVM Regression

Support vector machines were extended by Vapnik for regression The learning set of patterns is used to obtain a regression model that can be represented as a tube with radius e (Hyper – Parameter) fitted to the data. In the ideal case, SVM regression finds a function that maps all input data with a maximum deviation e from the target (experimental) values. In this case, all training points are located inside the regression tube. However, usually, it is not possible to fit all the patterns inside the tube and still have a meaningful model. For the general case, SVM regression considers that the error for patterns inside the tube is zero, whereas patterns situated outside the regression tube have an error that increases when the distance to the tube margin increases

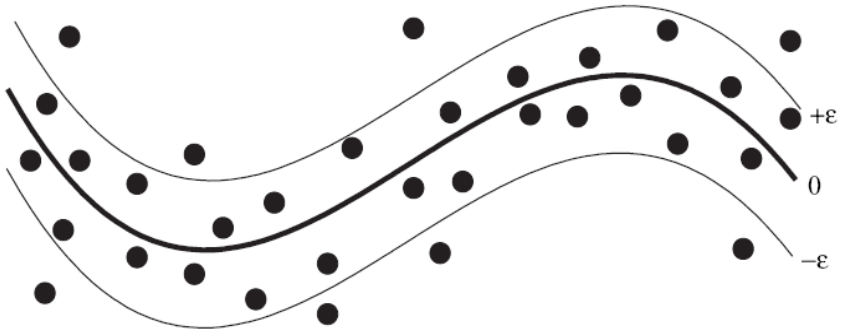
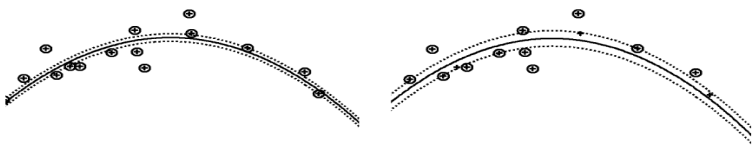


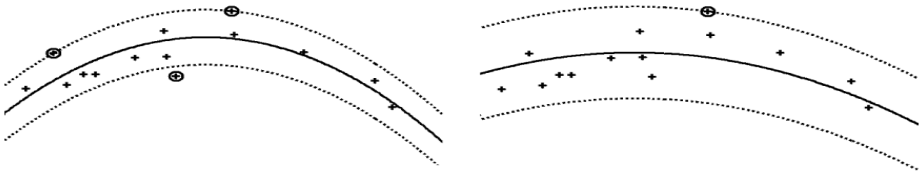
Fig. 7 the hypertube

A linear function is clearly inadequate for the dataset from the table below , so we will not present the SVMR model for the linear kernel. Patterns are represented by +, and support vectors are represented as + inside a circle. The SVM hyperplane is drawn with a continuous line, whereas the margins of the SVM regression tube are represented by dotted lines. Several experiments with different kernels showed that the degree 2 polynomial kernel offers a good model for this dataset, and we decided to demonstrate the influence of the tube radius e for this kernel. When the e parameter is too small, the diameter of the tube is also small forcing all patterns to be situated outside the tube. In this case, all patterns are penalized with a value that increases when the distance from the tube's margin increases.

No	Substituent X	ClogP	log $1/IC_{50}$
1	H	4.50	7.38
2	C_2H_5	4.69	7.66
3	$(CH_2)_2CH_3$	5.22	7.82
4	$(CH_2)_3CH_3$	5.74	8.29
5	$(CH_2)_4CH_3$	6.27	8.25
6	$(CH_2)_5CH_3$	6.80	8.06
7	$(CH_2)_7CH_3$	7.86	6.77
8	$CHMe_2$	5.00	7.70
9	$CHMeCH_2CH_3$	5.52	8.00
10	$CH_2CHMeCH_2CMe_3$	7.47	7.46
11	CH_2-cy-C_3H_5	5.13	7.82
12	CH_2CH_2-cy-C_6H_{11}	7.34	7.75
13	$CH_2COOCH_2CH_3$	4.90	8.05
14	$CH_2CO_2CMe_3$	5.83	7.80
15	$(CH_2)_5COOCH_2CH_3$	5.76	8.01
16	$CH_2CH_2C_6H_5$	6.25	8.51

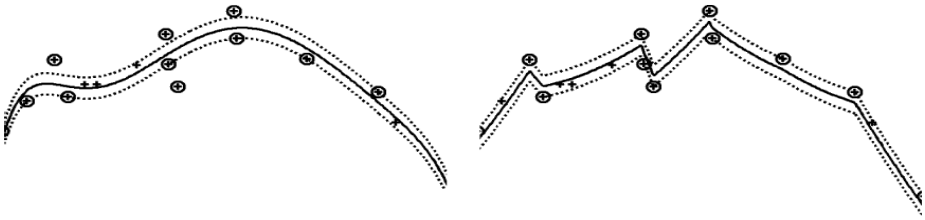


$\epsilon = 0.05$ $\epsilon = 0.1$



ϵ = 0.3 ϵ = 0.5

We can see how the variation of the radius changes the curvature



Polynomial Degree= 10 ϵ = 0.1 Radial Basis Function γ = 0.5 ϵ = 0.1

Using more complex kernel, the shape of the HyperTube changes a lot.

2.2.1 ϵ-Support Vector Regression (ϵ-SVR)

Given a set of data points, $\{(x_1; z_1), \ldots, (x_l; z_l)\}$, such that $x_i \in R^n$ is an input and $z_i \in R^1$ is a target output, the standard form of support vector regression (Vapnik, 1998) is:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i + C\sum_{i=1}^{l}\xi_i^* \\ \text{subject to} \quad & \mathbf{w}^T\phi(\mathbf{x}_i) + b - z_i \le \epsilon + \xi_i, \\ & z_i - \mathbf{w}^T\phi(\mathbf{x}_i) - b \le \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \ge 0, i = 1, \ldots, l. \end{aligned}$$

The dual is:

$$\begin{aligned} \min_{\alpha,\alpha^*} \quad & \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} z_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0, 0 \le \alpha_i, \alpha_i^* \le C, i = 1, \ldots, l, \\ & \text{where } Q_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j). \end{aligned}$$

2.2.2 ν-Support Vector Regression (ν -SVR)

Similar to ν-SVC, for regression, Scholkopf et al. (2000) use a parameter ν to control the number of support vectors. However, unlike ν -SVC, where ν replaces with C, here ν replaces with the parameter ϵ of ϵ-SVR. The primal form is:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*,\epsilon} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C(\nu\epsilon + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)) \\ \text{subject to} \quad & (\mathbf{w}^T\phi(\mathbf{x}_i) + b) - z_i \leq \epsilon + \xi_i, \\ & z_i - (\mathbf{w}^T\phi(\mathbf{x}_i) + b) \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, i = 1, \ldots, l, \ \epsilon \geq 0. \end{aligned}$$

And the dual is:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T Q(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \mathbf{z}^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\ \text{subject to} \quad & \mathbf{e}^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0, \ \mathbf{e}^T(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) \leq C\nu, \\ & 0 \leq \alpha_i, \alpha_i^* \leq C/l, \qquad i = 1, \ldots, l, \end{aligned}$$

2.3 SVM Practical Rules

The practice and expertise in the machine learning models, such as SVM, are important factors, coming from a long training and experience within their use in scientific experiments. The speed and effectiveness of the results strongly depend on these factors. Unfortunately there are no magic ways to a priori indicate the best configuration of internal parameters, involving network topology and learning algorithm.
But in some cases a set of practical rules to define best choices can be taken into account.

2.3.1 Scaling

Scaling before applying SVM is very important. Part 2 of Sarle's Neural Networks FAQ Sarle (1997) explains the importance of this and most of considerations also apply to SVM. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. We recommend linearly scaling each attribute to the range [1;+1] or [0; 1].
Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the rst attribute of training data from [10;+10] to [1;+1]. If the first attribute of testing data lies in the range [11;+8], we must scale the testing data to [1.1;+0.8].

2.3.2 Grid Search

We recommend a grid-search on the parameters there're two reasons for that:
One is that, psychologically, we may not feel safe to use methods which avoid doing an exhaustive parameter search by approximations or heuristics. The other reason is that the computational time required to find good parameters by grid-search is not much more than that by advanced methods since there are only few parameters

3 Use of the web application model

The Support Vector Machines are a very common supervised machine learning architectures used in many application fields. It is especially related to classification and regression problems, and in DAME it is designed to be associated with such two functionality domains. The description of these two functionalities is reported in the Reference Manual [A18], available from webapp menu or from the 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.

3.1 Use Cases

For the user the SVM system offers four use cases:

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

As described in [A19] a supervised machine learning model like SVM requires different use cases, well ordered in terms of execution sequence. A typical complete experiment with this kind of models consists in the following steps:

1. **Train** the network with a dataset as input, containing both input and target features; then store as output the final weight matrix (best configuration of network weights);
2. **Test** the trained network, in order to verify training quality (it is also included the validation step, available for some models). The same training dataset or a mix with new patterns can be used as input;
3. **Run** the trained and tested network with datasets containing ONLY input features (without target ones). In this case new or different input data are encouraged, because the Run use case implies to simply execute the model, like a generic static function.

The **Full** use case includes Train and Test cases. It can be executed as an alternative to the sequence of the two use cases. In this sense it is not to be considered as a single step of the sequence.

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): 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;

For training and test cases a correct dataset file must contain both input and target features (columns), with input type as the first group and target type as the final group.

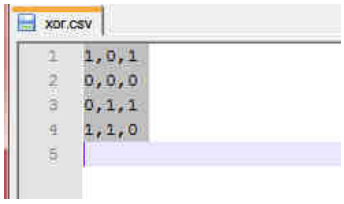


Fig. 8 - The content of the xor.csv file used as input for training/test use cases

As shown in Fig. 8, the xor.csv file for training/test uses cases has 4 patterns (rows) of 2 input features (first two columns) and one target feature (third column). The target feature is not an input information but the desired output to be used in the comparison (calculation of the error) with the model output during a training/test experiment.

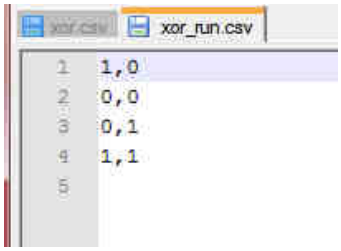


Fig. 9 - The content of the xor_run.csv file used as input for Run use case

In Fig. 9, the xor_run.csv file is shown, valid only for Run use case experiments. It is the same of xor.csv except for the target column that is not present. This file can be also generated by the user starting from the xor.csv. As detailed in the GUI user Guide [A19], the user may in fact use the Dataset Editor options of the webapp to manipulate and build datasets starting from uploaded data files.

3.3 Output

In terms of output, different files are obtained, depending on the specific use case of the experiment.
In the case of **classification** functionality, the following output files are obtained in all use cases:

TRAIN	TEST	FULL	RUN
Svm_TRAIN.log	Svm_TEST_mse	Svm_FULL_mse	Svm_RUN_stats
Svm_TRAIN_weights	Svm_TEST_stats	Svm_FULL_stats	Svm_RUN_output.dat
Svm_TRAIN_params.xml	Svm_TEST_output.dat	Svm_FULL_output.dat	Svm_RUN.log
	Svm_TEST.log	Svm_FULL.log	Svm_RUN_params.xml
	Svm_FULL_confusionMatrix	Svm_FULL_weights	
	Svm_TEST_params.xml	Svm_FULL_confusionMatrix	
		Svm_FULL_params.xml	

Tab. 1 – output file list in case of classification type experiments

In the case of **regression** functionality, the following output files are obtained in all use cases:

TRAIN	TEST	FULL	RUN
Svm_TRAIN.log	Svm_TEST_mse	Svm_FULL_mse	Svm_RUN_stats
Svm_TRAIN_weights	Svm_TEST_stats	Svm_FULL_stats	Svm_RUN_output.dat
Svm_TRAIN_params.xml	Svm_TEST_output.dat	Svm_FULL_output.dat	Svm_RUN.log
	Svm_TEST.log	Svm_FULL.log	Svm_RUN_params.xml
	Svm_TEST_outputPlot.jpeg	Svm_FULL_weights	
	Svm_TEST_params.xml	Svm_FULL_outputPlot.jpeg	
		Svm_FULL_params.xml	

Tab. 2 – output file list in case of regression type experiments

3.4 Train Use case

In the use case named "**Train**", the software provides the possibility to train the SVM. The user will be able to adjust parameters, 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).

3.4.1 Regression with SVM – Train Parameter Specifications

In the case of Regression_SVM with Train use case, the help page is at the address:
http://dame.dsf.unina.it/svm_help.html#regr_train

- **Input 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 include both input and target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be one of the types allowed by the application (ASCII, FITS, CSV, VOTABLE).

- **kernel type**

  This is the kernel type selection parameter. It defines the kind of the projection function used to define the support vectors in the parameter space. Take care of this choice. If left empty, the default is radial basis function.

  - **0** linear: u'*v
  - **1** polynomial: (gamma*u'*v + coef0)^degree
  - **2** radial basis function: exp(-gamma*|u-v|^2)
  - **3** sigmoid: tanh(gamma*u'*v + coef0)

- **svm type**

  this parameter is a field required!

  It indicates the possible choice between two different regression model engines:

  - **3** epsilon-SVR
  - **4** nu-SVR

- **gamma**

  It is one of the parameters included into the kernel definition function. It can be a positive real number. If left empty, the default value is 0.

- **degree**

  It is one of the parameters included into the kernel definition function. If left empty, the default value is 3

- **coeff 0**

  This is one of the kernel parameters. If left empty, the default value is 0

- **error tolerance**

  This is the threshold of the learning loop. This is the stopping criteria of the algorithm.

  If left empty the default value is 0.001

- **C**

  This is the penalty parameter of the epsilon-SVR model engine. If left empty, its default is 1.

- **nu**

  It is the nu-SVC and one-class SVM model engine parameter. If left empty the default value is 0.5

- **epsilon**

  Loss function in the epsilon-SVR. If left empty the default value is 0.001

- **shrinking**

  It is a flag indicating whether to use or not the shrinking heuristics, to speed up the algorithm. The default value is 1 (enabled)

3.4.2 Classification with SVM – Train Parameter Specifications

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

- **Input 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 include both input and target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be one of the types allowed by the application (ASCII, FITS, CSV, VOTABLE).

- **kernel type**

  This is the kernel type selection parameter. It defines the kind of the projection function used to define the support vectors in the parameter space. Take care about this choice. If left empty, the default is radial basis function. Following options are:

  - **0** linear: u'*v

- **1** polynomial: (gamma*u'*v + coef0)^degree
- **2** radial basis function: exp(-gamma*|u-v|^2)
- **3** sigmoid: tanh(gamma*u'*v + coef0)

- **svm type**

  It indicates the possible choice between three different model engines:

  - **0** C-SVC
  - **1** nu-SVC
  - **2** one-class SVM

  First two types are two different implementation types with the same behavior. Third one is particularly useful for the outliers search. The default is 0 (C-SVC).

- **gamma**

  It is one of the parameters included into the kernel definition function. It can be a positive real number. If left empty, the default value is 0.

- **degree**

  It is one of the parameters included into the kernel definition function. If left empty, the default value is 3

- **coeff 0**

  This is one of the kernel parameters. If left empty, the default value is 0

- **error tolerance**

  This is the threshold of the learning loop. This is the stopping criteria of the algorithm.

  If left empty the default value is 0.001

- **C**

  This is the penalty parameter of the C-SVC model engine. If left empty, its default is 1.

- **nu**

  It is the nu-SVC and one-class SVM model engine parameter. If left empty the default value is 0.5

- **weight**

This parameter must be selected only in case of C-SVC choice. It represents the weight of the stopping penalty parameter.

It is a multiplicative factor for the penalty parameter C (C*weight).

If users leaves empty this parameter field, the default value is set to 1

- **shrinking**

  It is a flag indicating whether to use or not the shrinking heuristics, to speed up the algorithm. The default value is 1 (enabled)

3.5 Test Use case

In the use case named "**Test**", the software provides the possibility to test the SVM. The user will be able to use already trained SVM 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).

3.5.1 Regression with SVM – Test Parameter Specifications

In the case of Regression_SVM with Test use case, the help page is at the address:
http://dame.dsf.unina.it/svm_help.html#regr_test

- **Input Dataset**

  this parameter is a field required!

  Dataset file as input. It is a file containing all input columns and the single target column.

  It must have the same number of input and target columns as for the training input file.

  For example, it could be the same dataset file used as the training input file.

- **Model File**

  this parameter is a field required!

It 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. The extension of such a file is usually .model

3.5.2 Classification with SVM – Test Parameter Specifications

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

- **Input Dataset**

  this parameter is a field required!

  Dataset file as input. It is a file containing all input columns and the single target column.

  It must have the same number of input and target columns as for the training input file.

  For example, it could be the same dataset file used as the training input file.

- **Model File**

  this parameter is a field required!

  It 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. The extension of such a file is usually .model

3.6 Run Use case

In the use case named "**Run**", the software provides the possibility to run the SVM. The user will be able to use already trained and tested SVM models, their weight configurations, to execute the normal experiments on new datasets.

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 Regression with SVM – Run Parameter Specifications

In the case of Regression_SVM with Run use case, the help page is at the address:
http://dame.dsf.unina.it/svm_help.html#regr_run

- **Input Dataset**

  this parameter is a field required!

Dataset file as input. It is a file containing only input columns (without target column).

It must have the same number of input columns as for the training input file.

For example, it could be the same dataset file used as the training input file, BUT WITHOUT THE TARGET COLUMN.

- **Model File**

  this parameter is a field required!

  It 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. The extension of such a file is usually .model

3.6.2 Classification with SVM – Run Parameter Specifications

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

- **Input Dataset**

  this parameter is a field required!

  Dataset file as input. It is a file containing only input columns (without target column).

  It must have the same number of input columns as for the training input file.

  For example, it could be the same dataset file used as the training input file, BUT WITHOUT THE TARGET COLUMN.

- **Model File**

  this parameter is a field required!

  It 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. The extension of such a file is usually .model

3.7 Full Use case

In the use case named "**Full**", the software provides the possibility to perform a complete sequence of train and test cases with the SVM.

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 Regression with SVM – Full Parameter Specifications

In the case of Regression_SVM with Full use case, the help page is at the address:
http://dame.dsf.unina.it/svm_help.html#regr_full

- **Training Set**

  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 include both input and target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be one of the types allowed by the application (ASCII, FITS, CSV, VOTABLE).

- **Test Set**

  this parameter is a field required!

  Dataset file as test input. It is a file containing all input columns and the single target column.

  It must have the same number of input and target columns as for the training input file.

  For example, it could be the same dataset file used as the training input file.

- **kernel type**

  This is the kernel type selection parameter. It defines the kind of the projection function used to define the support vectors in the parameter space. Take care of this choice. If left empty, the default is radial basis function.

  - **0** linear: u'*v
  - **1** polynomial: (gamma*u'*v + coef0)^degree

- **2** radial basis function: exp(-gamma*|u-v|^2)
- **3** sigmoid: tanh(gamma*u'*v + coef0)

- **svm type**

  this parameter is a field required!

  It indicates the possible choice between two different regression model engines:

  - **3** epsilon-SVR
  - **4** nu-SVR

- **gamma**

  It is one of the parameters included into the kernel definition function. It can be a positive real number. If left empty, the default value is 0.

- **degree**

  It is one of the parameters included into the kernel definition function. If left empty, the default value is 3

- **coeff 0**

  This is one of the kernel parameters. If left empty, the default value is 0

- **error tolerance**

  This is the threshold of the learning loop. This is the stopping criteria of the algorithm.

  If left empty the default value is 0.001

- **C**

  This is the penalty parameter of the epsilon-SVR model engine. If left empty, its default is 1.

- **nu**

  It is the nu-SVC and one-class SVM model engine parameter. If left empty the default value is 0.5

- **epsilon**

  Loss function in the epsilon-SVR. If left empty the default value is 0.001

- **shrinking**

  It is a flag indicating whether to use or not the shrinking heuristics, to speed up the algorithm. The default value is 1 (enabled)

3.7.2 Classification with SVM – Full Parameter Specifications

In the case of Classification_SVM with Full use case, the help page is at the address:
http://dame.dsf.unina.it/svm_help.html#class_full

- **Training Set**

  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 include both input and target columns, where each row is an entire pattern (or sample of data). The format (hence its extension) must be one of the types allowed by the application (ASCII, FITS, CSV, VOTABLE).

- **Test Set**

  this parameter is a field required!

  Dataset file as test input. It is a file containing all input columns and the single target column.

  It must have the same number of input and target columns as for the training input file.

  For example, it could be the same dataset file used as the training input file.

- **kernel type**

  This is the kernel type selection parameter. It defines the kind of the projection function used to define the support vectors in the parameter space. Take care of this choice. If left empty, the default is radial basis function.

  - **0** linear: u'*v
  - **1** polynomial: (gamma*u'*v + coef0)^degree
  - **2** radial basis function: exp(-gamma*|u-v|^2)
  - **3** sigmoid: tanh(gamma*u'*v + coef0)

- **svm type**

  It indicates the possible choice between three different model engines:

  - **0** C-SVC
  - **1** nu-SVC
  - **2** one-class SVM

First two types are two different implementation types with the same behavior. Third one is particularly useful for the outliers search. The default is 0 (C-SVC).

- **gamma**

  It is one of the parameters included into the kernel definition function. It can be a positive real number. If left empty, the default value is 0.

- **degree**

  It is one of the parameters included into the kernel definition function. If left empty, the default value is 3

- **coeff 0**

  This is one of the kernel parameters. If left empty, the default value is 0

- **error tolerance**

  This is the threshold of the learning loop. This is the stopping criteria of the algorithm.

  If left empty the default value is 0.001

- **C**

  This is the penalty parameter of the C-SVC model engine. If left empty, its default is 1.

- **nu**

  It is the nu-SVC and one-class SVM model engine parameter. If left empty the default value is 0.5

- **weight**

  This parameter must be selected only in case of C-SVC choice. It represents the weight of the stopping penalty parameter.

  It is a multiplicative factor for the penalty parameter C (C*weight).

  If users leaves empty this parameter field, the default value is set to 1

- **shrinking**

  It is a flag indicating whether to use or not the shrinking heuristics, to speed up the algorithm. The default value is 1 (enabled)

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.

4.1 Classification Seyfert 1 vs Seyfert 2

This is a classification problem in the dataset we have 2 classes (stated as 1 and 0) and 13 features.

Seyferts were first classified as Type 1 or 2, depending upon whether the spectra show both narrow and broad emission lines (Type 1), or only narrow lines (Type 2). They are now given a fractional classification depending upon the relative strengths of the narrow and broad components (e.g. Type 1.5 or Type 1.9).[4] It is believed that Type 1 and Type 2 galaxies are in essence the same, and they only differ due to the angle at which they are observed. This is known as Seyfert Unification theory. In Type 2 Seyferts it is believed that the broad component is obscured by dust and/or by our viewing angle on the galaxy. In some Type 2 Seyfert galaxies, the broad component can be observed in polarized light; it is believed that light from the broad-line region is scattered by a hot, gaseous halo surrounding the nucleus, allowing us to view it indirectly. This effect was first discovered by Antonucci and Miller in the Type 2 Seyfert NGC 1068.

AGNss were classified by Sorrentino et al. (The environment of active galaxies in the SDSS-DR4) as Sy1 if FWHM(Hα) > 1.5 FWHM([OIII]λ5007), or as Sy2 otherwise. We also classified as Sy1 all the emission-line galaxies having at least Hα and [OIII]λ5007 emission-lines with FWHM(Hα) > 1200 km s−1 and FWHM([OIII]λ5007) < 800 km s−1, independent of line ratios: these limits were empirically found by looking at the distribution of the FWHMs and examining the spectra.

Our dataset is the join of the public catalogue
http://www.mpa-garching.mpg.de/SDSS/DR4/Data/agncatalogue.html
that contains the subset of 88178 emission line galaxies from the stellar mass catalogue that are classified as AGN as described in Kauffmann et al 2003, MNRAS, 346, 1055, "The host galaxies of active galactic nuclei" and the one obtained by Sorrentino et al.
The features columns are:
1. petroR50 u
2. petroR50 g
3. petroR50 r
4. petroR50 i
5. petroR50 z
6. concentration index r
7. z phot corr
8. fibermag r
9. (u − g)dered)
10. (g − r)dered)
11. (r − i)dered)
12. (i − z)dered)
13. dered r

All the previous columns are extracted from the photometric SDSS catalogues with the exception of the column 7 (z phot corr) that is a photometric evaluation of the Redshift obtained By D'Abrusco et al. *(Mining the SDSS archive. I. Photometric redshifts in the nearby universe., arXiv:astro-ph/0703108v2 9 Mar, (2007))*

The target column is a flag, 1 if an object is a Type I object (according Sorrentino et al.) or a Type 2 object.

As first case, we will use the SVM model associated to the classification functionality.
The starting point is to create a new workspace, named **SVMExp** and to populate it by uploading the file:

- **seyfert.csv**: CSV full dataset containing all the objects with features and target vector

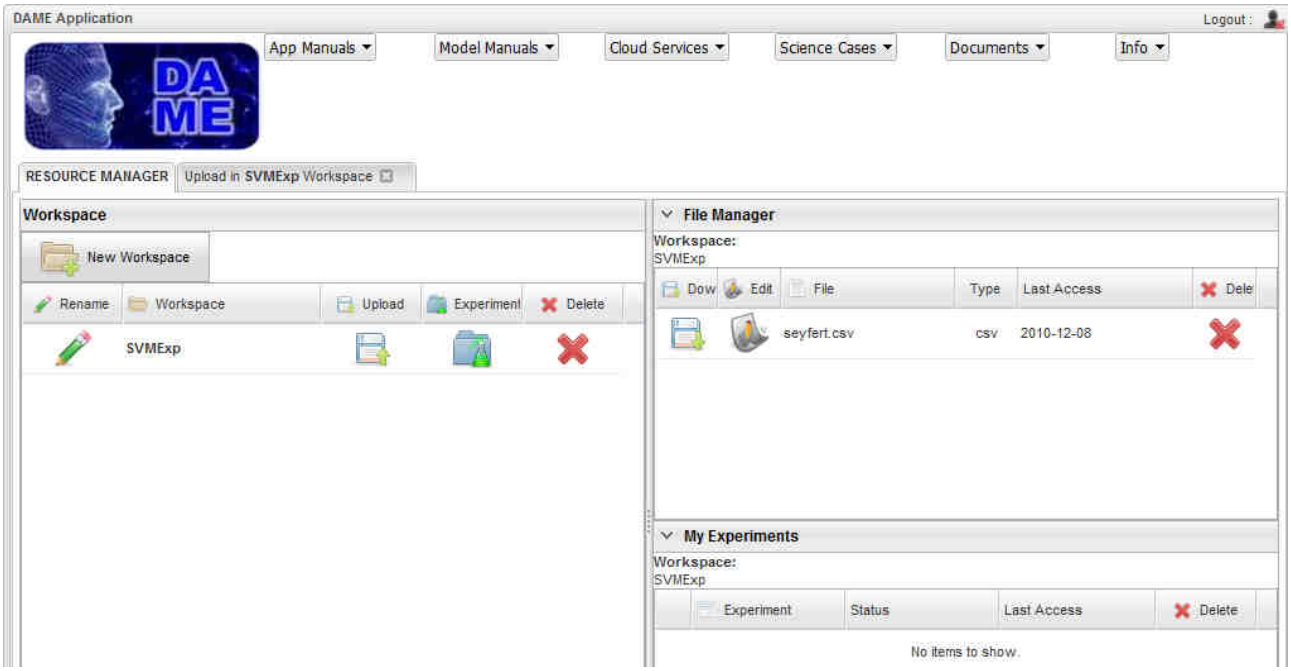


Fig. 10 – The starting point, with a Workspace (SVMExp) created and the data file uploaded

4.1.1 Classification SVM – Creation of Train, Test and Run set

In order to make this experiment we need to create the datasets for the various use cases so let's press on the edit button on the left of the seyfert.csv file in the File Manager panel

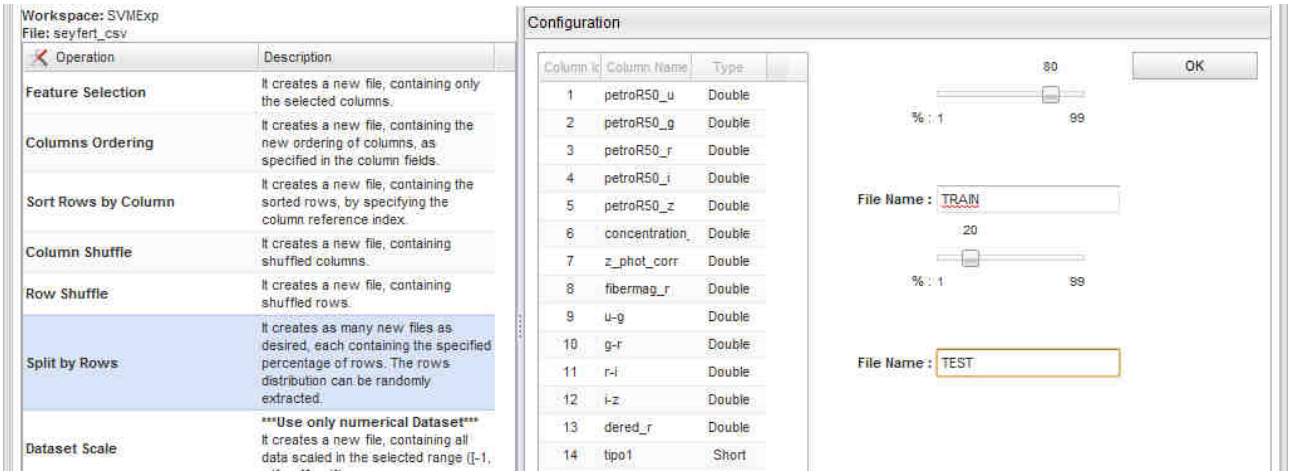


Fig. 11 - Split by Rows

We need to choose the "Split by Rows" operation in order to obtain two subsets, in the right panel then we can configure this operation, we choose for the first subset the name TRAIN and 80% of the dimension, the second subset will be named TEST and take the remaining 20%

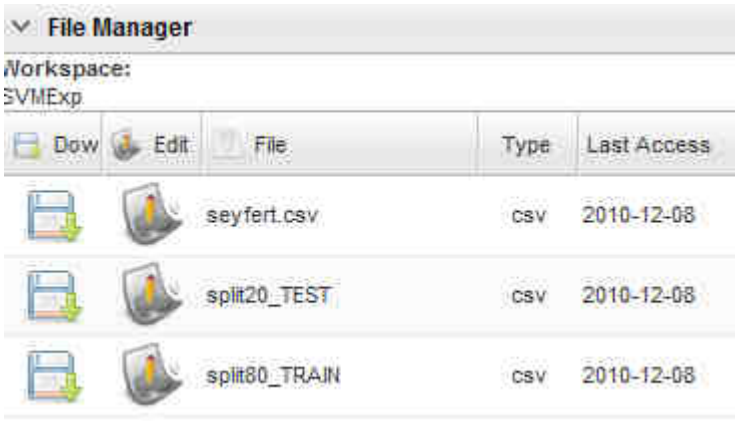


Fig. 12 - Splitted datasets

Now in the File Manager we can find our two subsets, we need another file for the Run case, we can take the Test set without the last column (target vector), so press on the edit button on the left of split20_TEST

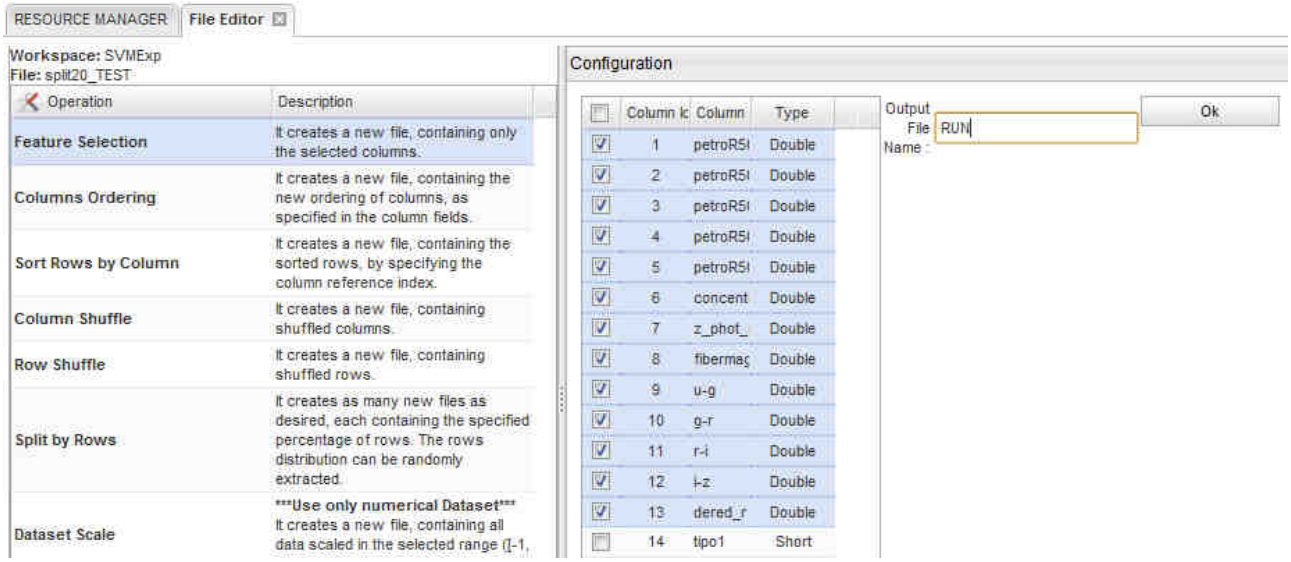


Fig. 13 - Feature selection

This time we need to make a Feature Selection, so we select all the column with the exception of the last one and give the RUN name to the file and here we can see the new File Manager:

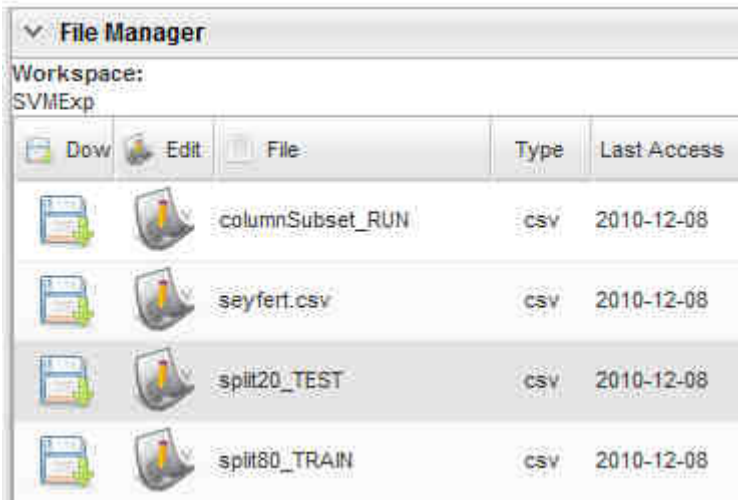


Fig. 14 - The produced Run Set

4.1.2 Classification SVM – Train use case

Let suppose we create an experiment named **seyfertTrain** and we want to configure it. After creation, the new configuration tab is open. Here we select **Classification_SVM** as couple functionality-model of the current experiment and we select also **Train** as use case.

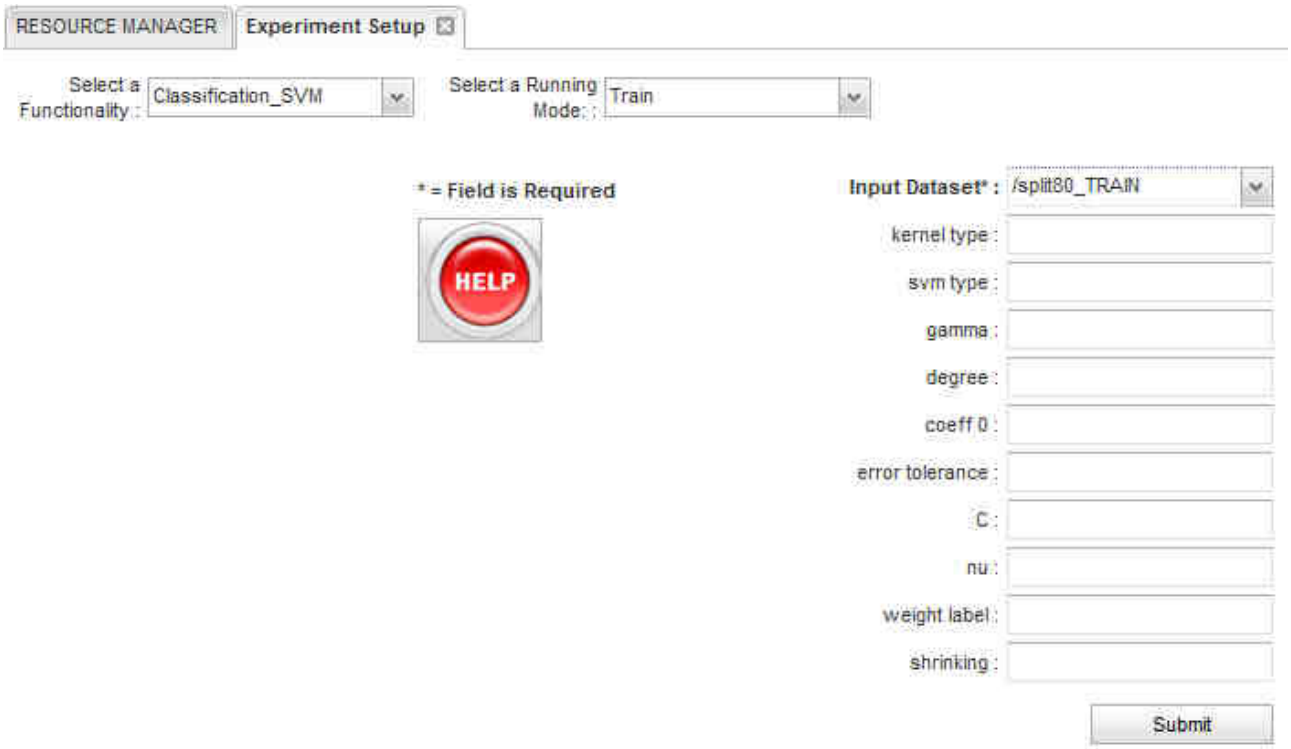


Fig. 15 – The seyfertTrain experiment configuration tab

Now we have to configure parameters for the experiment. In particular, we will leave empty the not required fields (labels without asterisk).
The meaning of the parameters for this use case are described in previous sections of this document. As alternative, you can click on the Help button to obtain detailed parameter description and their default values directly from the webapp.
We need just to give split80_TRAIN as input dataset.

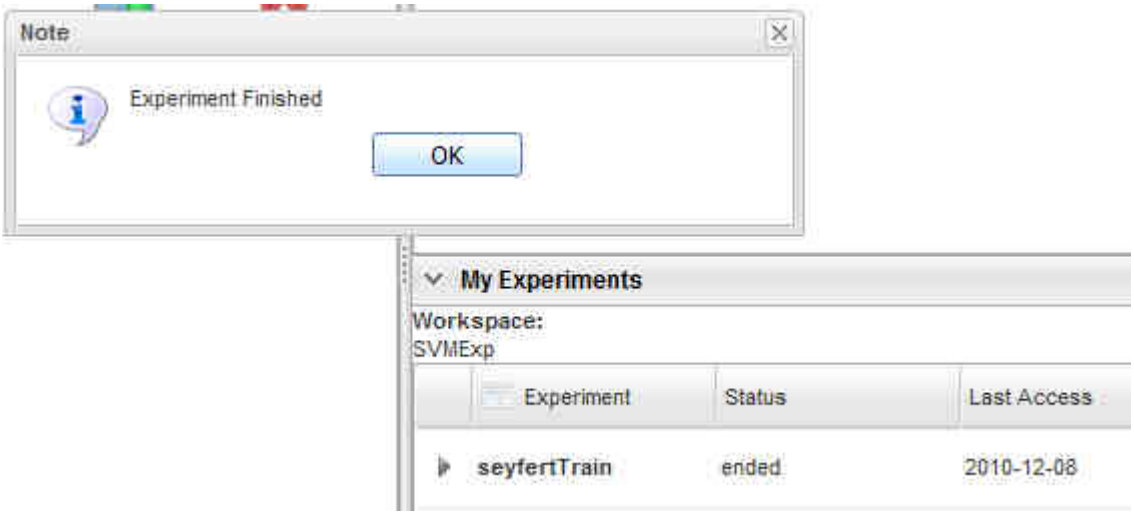


Fig. 16 – The seyfertTrain experiment status after submission

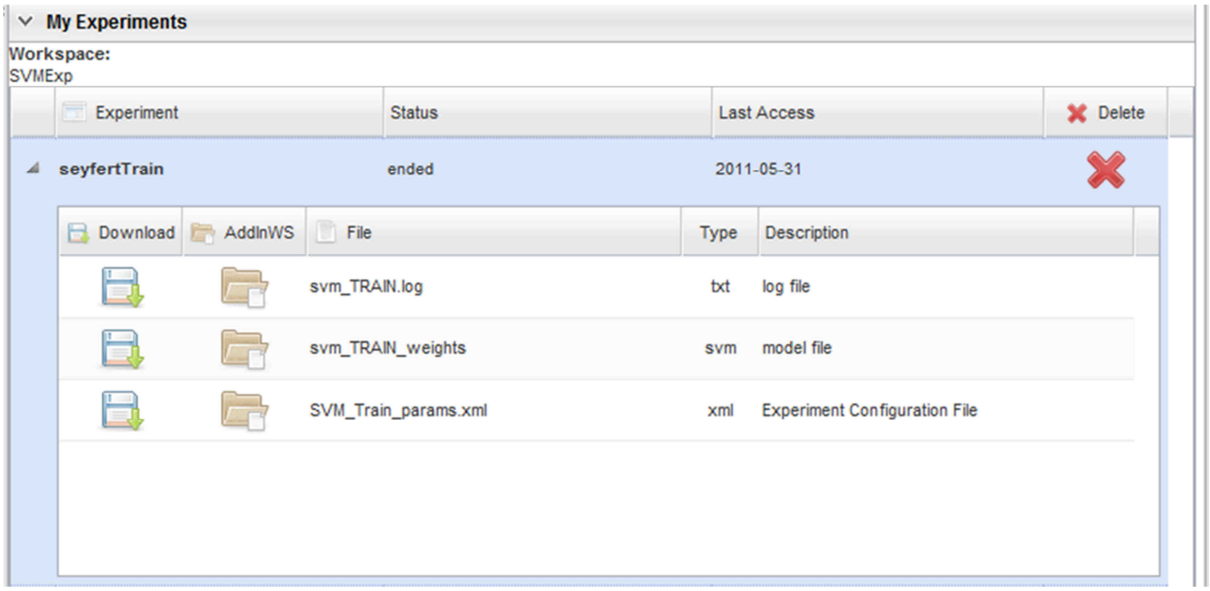


Fig. 17 – The seyfertTrain experiment output files

The content of output files, obtained at the end of the experiment (available when the status is “ended”) is shown in the following.
The file svm_TRAIN_weights contains the topology of the svm model and the “support vectors” retrieved by the algorithm:

```
svm_type c_svc
kernel_type rbf
gamma 0.07692307692307693
nr_class 2
total_sv 855
rho 0.878853313659399
label 0 1
probA -1.6591144414568746
probB 0.4112262539118618
nr_sv 427 428
SV
1.0 1:2.9987934 2:3.2648723 3:3.083676 4:2.7884176 5:2.583206
6:0.33331323 7:0.051167794 8:17.54045 9:1.456662999999999
10:0.7198779999999996 11:0.446237 12:0.29616500000000023 13:15.655557
1.0 1:2.9750006 2:3.7500892 3:3.6670167 4:3.4655538 5:3.3231199
6:0.3816999 7:0.080045864 8:18.14997 9:1.5674470000000014
10:0.8488170000000004 11:0.44479699999999944 12:0.3312640000000009
13:15.855777
1.0 1:1.2115157 2:1.15915 3:1.1400326 4:1.1477162 5:1.1264294
6:0.38862574 7:0.059009977 8:17.39233 9:1.5125390000000003
10:0.6746289999999995 11:0.34969300000000203 12:0.19137599999999821
13:16.670805
1.0 1:0.7758318 2:0.7151856 3:0.7481465 4:0.7117787 5:0.7009405
6:0.46411586 7:0.14799979 8:17.54169 9:2.2949289999999998 10:1.128456
11:0.39285899999999785 12:0.3726230000000008 13:17.438332
1.0 1:4.5174656 2:3.461495 3:3.0866225 4:2.7874756 5:2.3104877
6:0.30233362 7:0.072981484 8:17.71745 9:1.6488759999999978
10:0.8305710000000026 11:0.4135169999999988 12:0.29349700000000034
```

Fig. 18 - SVM File Model

4.1.3 Classification SVM – Test use case

The file svm_TRAIN_weights can be copied into the input file area (**File Manager**) of the workspace, in order to be re-used in future experiments (for example in this case the test use case). This is because it represents the stored brain of the network, trained to calculate the seyfert 1 – 2 separation function.

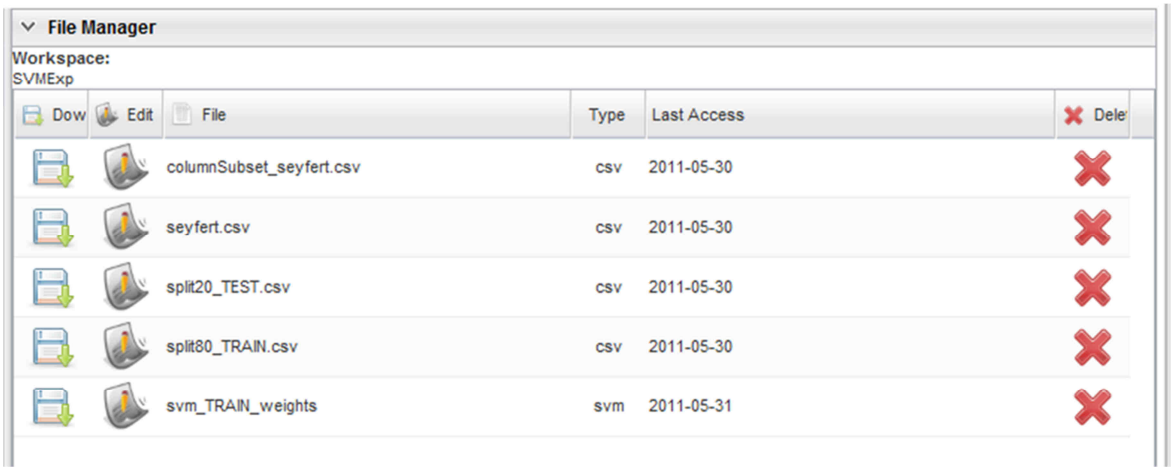


Fig. 19 – The file "svm_TRAIN_weights" copied in the WS input file area for next purposes

So far, we proceed to create a new experiment, named **seyfertTest**, to verify the training of the network. We will use the file split20_TEST that we create before through file editing options.

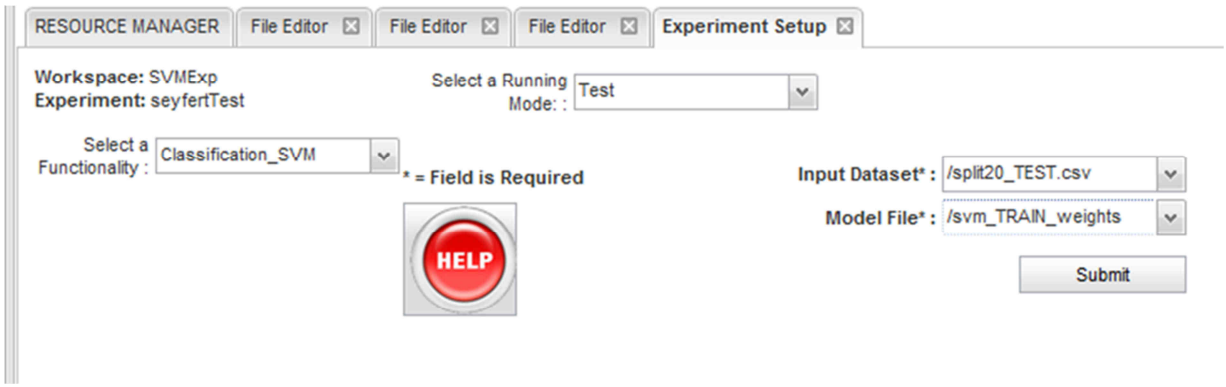


Fig. 20 – The seyfertTest experiment configuration tab (note "svm_TRAIN_weights" file inserted)

After execution, the experiment **seyfertTest** will show the output files available.

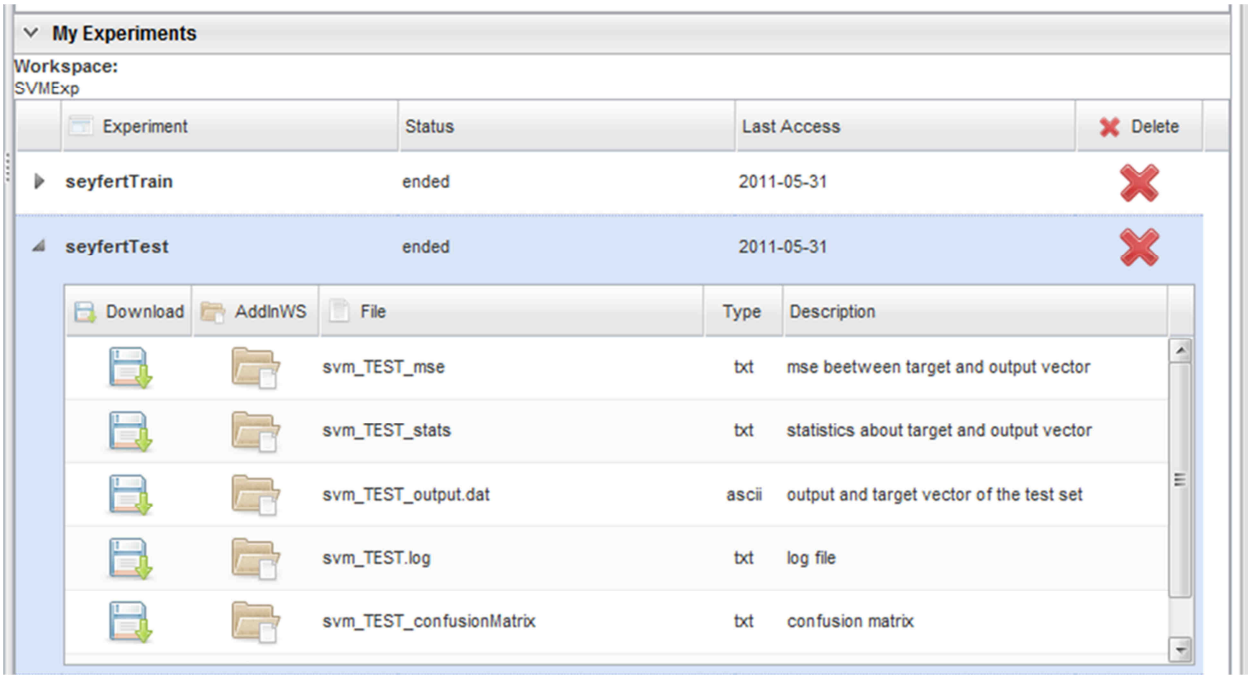


Fig. 21 – The seyfertTest experiment output files (file names may change)

4.1.4 Classification SVM – Full use case

If an automatic sequence of train and test use cases is desired, it is possible to execute an experiment by choosing Full as use case.
In this case, we create a new experiment, named **xorFull**, where we have to select parameters for both train and test use cases.
At the end, we obtain the output files of the experiment, in which both training and test outputs are present.

4.1.5 Classification SVM – Run use case

Usually after some train with different parameters (try for example to set in the train gamma=0.03125 and C=512.0001 and see what change in the final test output) the test will certificate to you if the SVM so trained solves your problem then you have to apply it to data for with you don't have the target vector in order to make new discovery, this is the Run use case, try to apply the model previously trained to the Run subset we obtained before.

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	MLP	Multi Layer Perceptron
CE	Cross Entropy	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
DAPL	Data Access & Process Layer	PI	Principal Investigator
DL	Data Layer	REDB	Registry & Database
DM	Data Mining	RIA	Rich Internet Application
DMM	Data Mining Model	SDSS	Sloan Digital Sky Survey
DMS	Data Mining Suite	SL	Service Layer
FITS	Flexible Image Transport System	SW	Software
FL	Frontend Layer	UI	User Interface
FW	FrameWork	URI	Uniform Resource Indicator
GRID	Global Resource Information Database	VO	Virtual Observatory
GUI	Graphical User Interface	XML	eXtensible Markup Language
HW	Hardware		

Tab. 3 – Abbreviations and acronyms

Reference & Applicable Documents

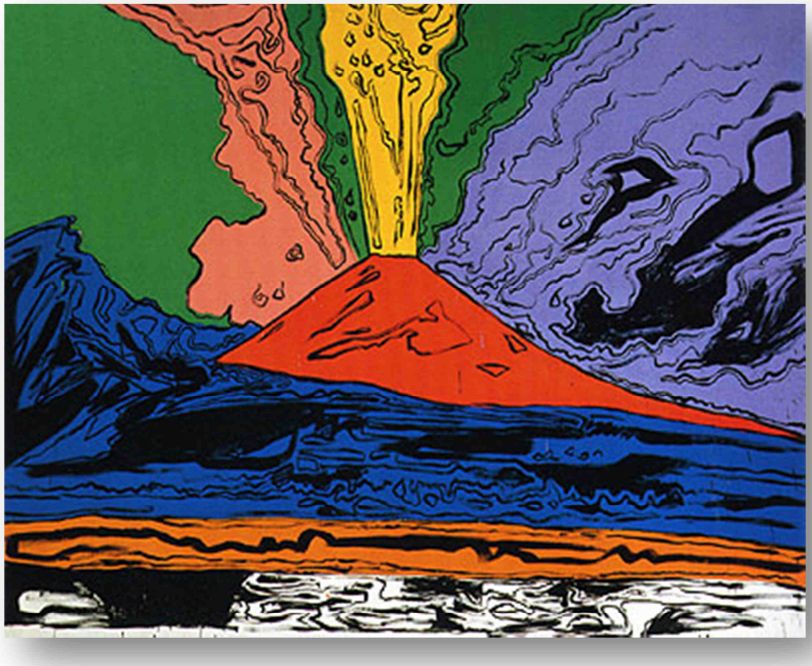
ID	Title / Code	Author	Date
R1	"The Use of Multiple Measurements in Taxonomic Problems", in Annals of Eugenics, 7, p. 179—188	Ronald Fisher	1936
R2	*Neural Networks for Pattern Recognition*. Oxford University Press, GB	Bishop, C. M.	1995
R3	*Neural Computation*	Bishop, C. M., Svensen, M. & Williams, C. K. I.	1998
R4	Data Mining Introductory and Advanced Topics, Prentice-Hall	Dunham, M.	2002
R5	Mining the SDSS archive I. Photometric Redshifts in the Nearby Universe. *Astrophysical Journal*, Vol. 663, pp. 752-764	D'Abrusco, R. et al.	2007
R6	*The Fourth Paradigm*. Microsoft research, Redmond Washington, USA	Hey, T. et al.	2009
R7	Artificial Intelligence, A modern Approach. Second ed. (Prentice Hall)	Russell, S., Norvig, P.	2003
R8	Pattern Classification, A Wiley-Interscience Publication, New York: Wiley	Duda, R.O., Hart, P.E., Stork, D.G.	2001
R9	Neural Networks - A comprehensive Foundation, Second Edition, Prentice Hall	Haykin, S.,	1999
R10	*A practical application of simulated annealing to clustering.* Pattern Recognition 25(4): 401-412	Donald E. Brown D.E., Huntley, C. L.:	1991
R11	*Probabilistic connectionist approaches for the design of good communication codes.* Proc. of the IJCNN, Japan	Babu G. P., Murty M. N.	1993
R12	*Approximations by superpositions of sigmoidal functions.* Mathematics of Control, Signals, and Systems, 2:303–314, no. 4 pp. 303-314	Cybenko, G.	1989

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	SVM_user_manual_VONEURAL-MAN-NA-0002-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	LIBSVM: a Library for Support Vector Machines	Chih-Chung Chang, Chih Jen Lin	02/10/2007
A14	A practical guide to support vector classification	C.-W. Hsu, C.-C. Chang, C.-J. Lin	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

Tab. 5 – Applicable Documents

__oOo__



DAME Program
"we make science discovery happen"