Quick-Start-Guide: Using the software tool 'solver' (aka 'uSort')

v1.0
rev. 3 19.08.2023

Copyright 2023 Daniel Sanders
URL: https://github.com/dsandersGit/GIT_Solver

Content

What it is:

Solver is a tool that uses machine learning approaches for learning how to predict different classes of samples based on sample describing variables/features (discrimintant analysis). The necessary data consist of a set of samples, each exhibiting a class-label (a classes descriptive word) and multiple determinable numeric variables. The following table sketches such a data set using an extract of the well known IRIS dataset (https://en.wikipedia.org/wiki/Iris_flower_data_set).

Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species / Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
7	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
7	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor

Fig. 1: Extract of the Iris data set

'Species/Class' is the class-label, the other columns are determinable characteristics (here: morphologic variations of the iris flower).
Aim of the machine learning process is the discovery of characteristic relations of variables that is linked to a specific class. Such relations will be called 'models'.
When models are determined, they can be applied on new, unclassified data to predict the respective class. Such models can be saved and applied later for classification.
'Solver' uses an approach of iteratively setting one class as 'target'. Machine learning then aims on the separation of all other class's samples

by randomly adjusting vectors, that are used in linear combination with the variables. If the result of the linear combination spatially displaces target class from other classes, the set of vectors are used again, if not, the last change is revoked (backpropagation). If no further separation is observed the respective vector set is the 'model' for classification of the respective target. Walking through all classes as target and repeating the full cylce results in a set of models (ensemble).
If sufficient samples are available, only a ratio of samples are used for the machine learning training. Untouched samples represent the test data. Their classification reveal the classification performance (generalization) and can be used for model pruning.

The machine learning training ultimatively generates a set of classification models in an ensemble. This can be saved as data file (*.ens) and can be equipped on new samples for estimating the respective class. For classification new samples, with the same variable selection, are loaded as data. Then an existing matching ensemble is loaded and applied on the samples for class estimation.

What you need to run the app:

i) *'uSort.jar'* is a JAVA runnable JAR file. For execution the JAR file must be opened with a JAVA runtime environment (JRE). This can be installed on a computer or the JAR file can be opened with JRE's 'javaw.<executable>' (*'solver' should be OS independent*). Free JRE's can be found online.

ii) Data. For a quick-start you should find the Iris data set together with the app. Use this to get a feeling.

If you want to try your own data, concern the following:
   – arrange your samples with class-label and variables in a line using your preferred spreadsheet app. A header is optional.
   – save as 'comma separated file' *.csv
   – class-label can be at any column position, only one class label is allowed. The class column is detected by the precence of non-number String.
   – number of variables should be smaller than number of samples. Otherwise any discriminant analysis will tend to 'overfit' and find models that does not generalize new unclassified samples. If possible, remove highly correlated variables (>redundant information). Spearman rank correlation could be beneficial.
   – minimum number of classes is 2

How to run:

Start the app

Start the app (double click on uSort.jar if JRE is installed, otherwise set the paths in cmd.exe, type <JRE PATH>\javaw.exe" -jar "<APP PATH>\ uSort.jar" %)
in case of problems: net search for 'run a java jar file'

Load data

A user interface pops up. Select the data set file you want to process for learning (menubar / iconbar right-hand-side). The |data|-tab will show selected features of the data set:
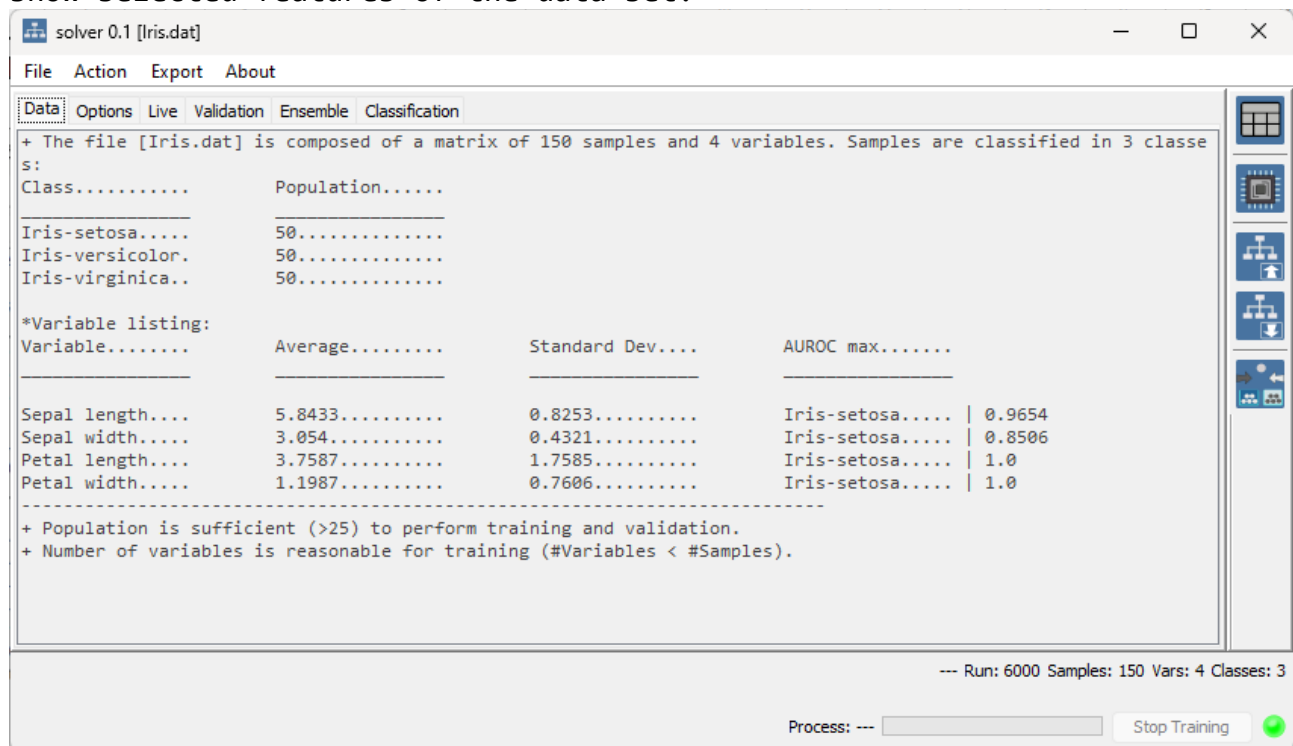

Fig. 2: |data|-tab when iris data set is loaded

Note: The 'AUROC max' (highest value for single class, AUROC - area under receiver operator characteristics) is only calculated for smaller sample / variable number due to calculation times. If an AUROC of 1 is found for variables they can be used to separate the stated class univariately.

Set options

Select the |Options|-tab:
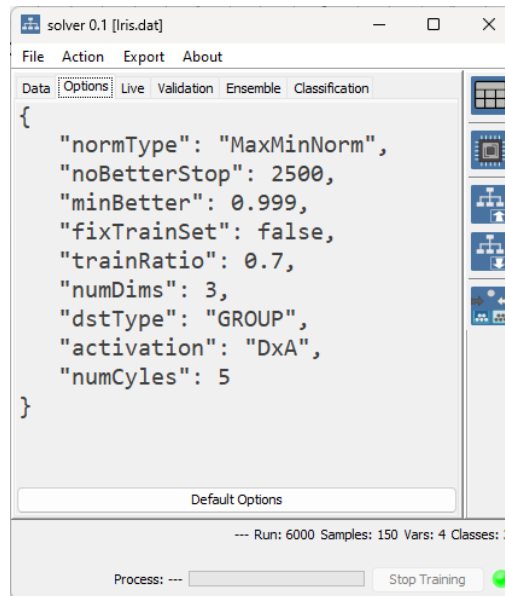
Fig. 3: |options|-tab [default]

Options affects the learning/training process. Important options are:
- **numCycles**: number of cycles that are trained. One cylce trains all classes. Together the gained models form an 'ensemble' where multiple models are used for classification. Higher number might raise accuracy.
- **noBetterStop**: learning/training will stop when no improvement of model accuracy/bonus is achieved. Number determines how many unsuccessful steps are tolerated before stopping the current target-learning. Higher number > longer training time, but might improve accuracy
- **numDims**: The used algorithm is multivariate, numDims sets the number of dimensions/neurons. Complex data / multiple classes might benefit from higher dimensionality.
- **fixTrainSet**: samples are split in training and testing subsets. Learning algorithm only uses the train set while the test set result indicates the generalization potential of models. A fixes train/test set can be selected 'true' for the whole ensemble training, or for each model an new train/testing subset is created.
- **trainRation**: determines the ratio of the number of training to testing samples. 0.7 > 70% of sample will be used for training, 30% for testing. Ration is applied for each class. For small sample numbers splitting train / testing is not recommended.
-

The default options should cover most tasks.

Change options in the text. For boolean options use true/false. Options are processed as JSON format and will be stored together with your ensemble.

4

Run a training

Switch to the |Live|-tab, then start 'train' (menubar / iconbar right-hand-side):
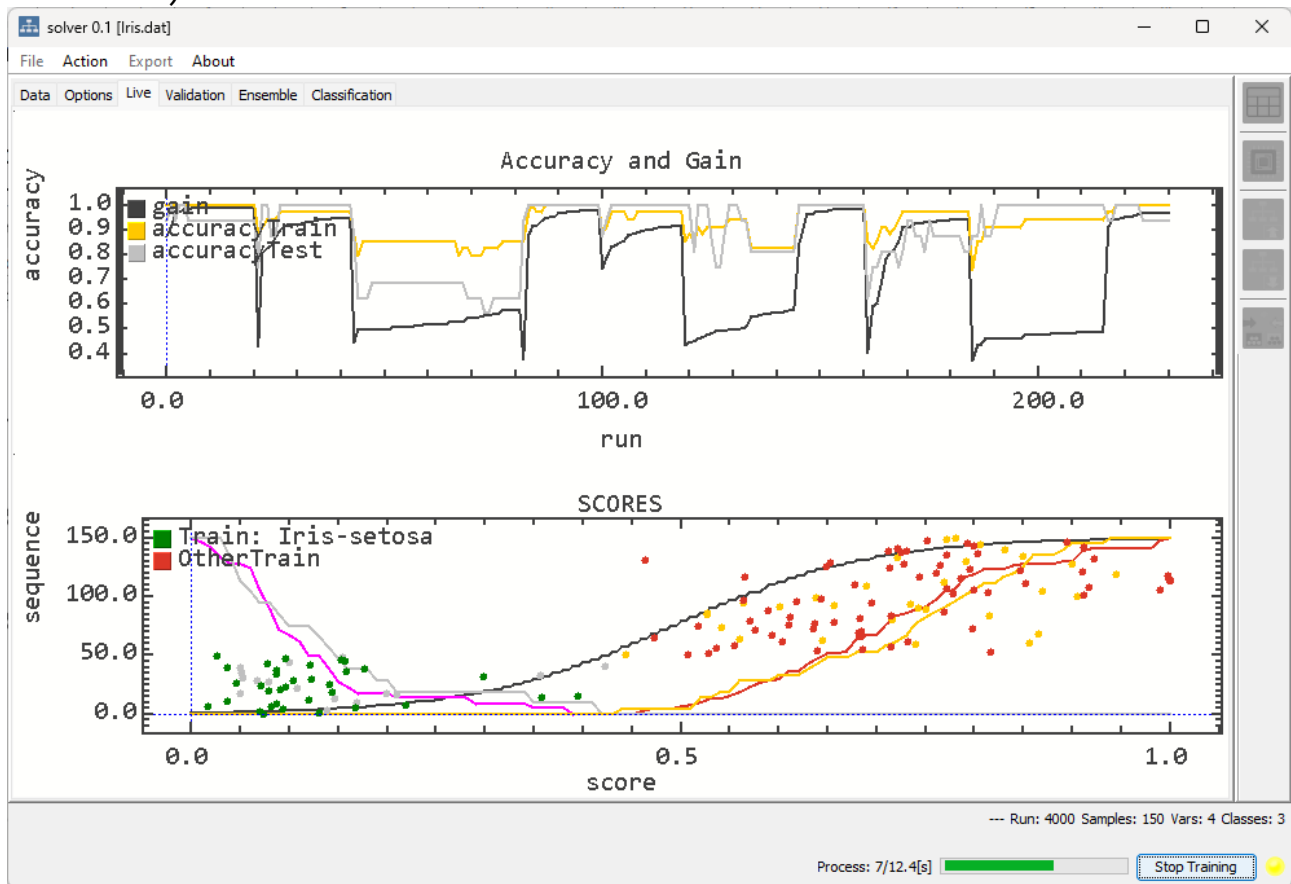

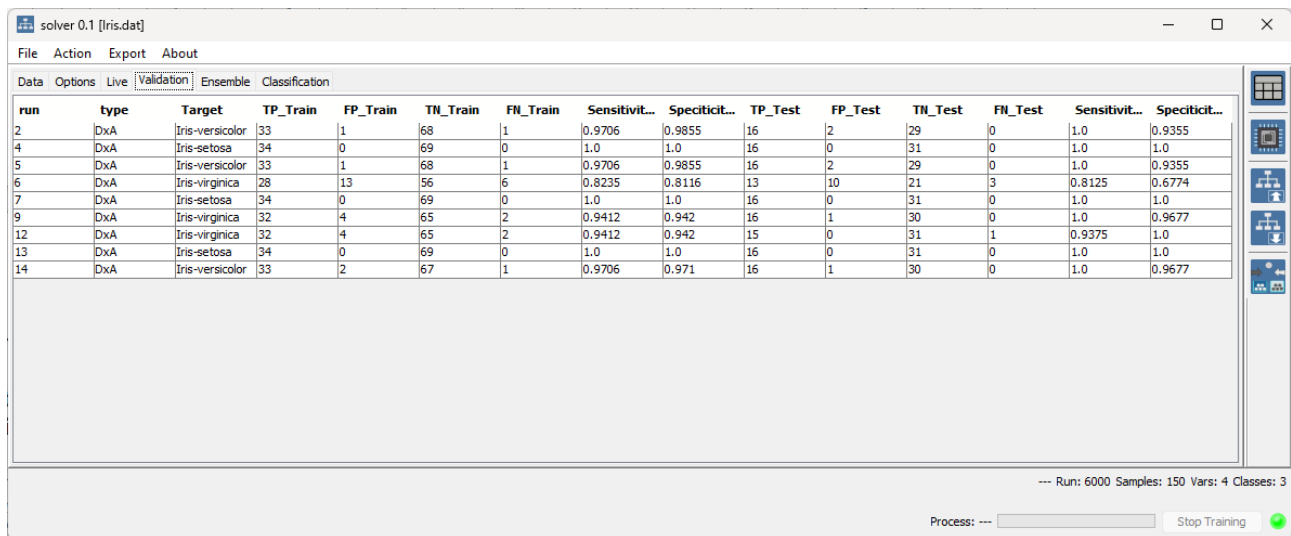Fig. 4: |live|-tab after running a training on iris data set

Learning / Training is visualized live. Top graph plots accuracy of the classification for both training and testing data on the run, furthermore the underlying algorithm's bonus is shown. Only improved bonus is plotted.
Well aligned 'accuracyTrain' and 'accuracyTest' indicates good generalization potential.

Bottom graph plots the samples sequence (number/sorting) on the 'score'. The score is a result of the discrimination algorithm. In each pass only one sample class is set as target, and the algorithm tries shifting the target to score 0 while pushing all other samples towards 1. The best split score is determined that separates target from non-target. When no more improvement is found the model is stored and the target will be set to the next class.

Next to |live|-tab progress can be followed looking at the |validation|-tab. Here, the detailed accuracy of each pass is listed live. Both training and testing samples are analysed on the model's performance by

true positive (TP) false positive (FP) true negative (TN) and false negative (FN) classification. Furthermore, the sensitivity and specificity of each pass is shown.

Fig. 5: |validation|-tab after running a training on iris data set

The training result

After finishing, the developed model-ensemble is accessible. For testing it can be applied on the current data set (menubar / iconbar right-hand-side). Here, the whole ensemble, not individual models, are used to classify samples. Respective classes numbers indicate the 'frequency' of positive classification. The highest frequency determines the overall classification:

Fig. 6: |classification|-tab after classification

Column 'classification' is the estimated classification. The model-ensemble can be viewed in the |ensemble|-tab. This is the instruction on how to process new, unclassified sample's variables, towards estimating the class belonging.
Save the ensemble for further classification jobs.

Run a classification

Load a data set of unclassified samples. Load a classification ensemble.
Variables in both must be equal. Run a classification.
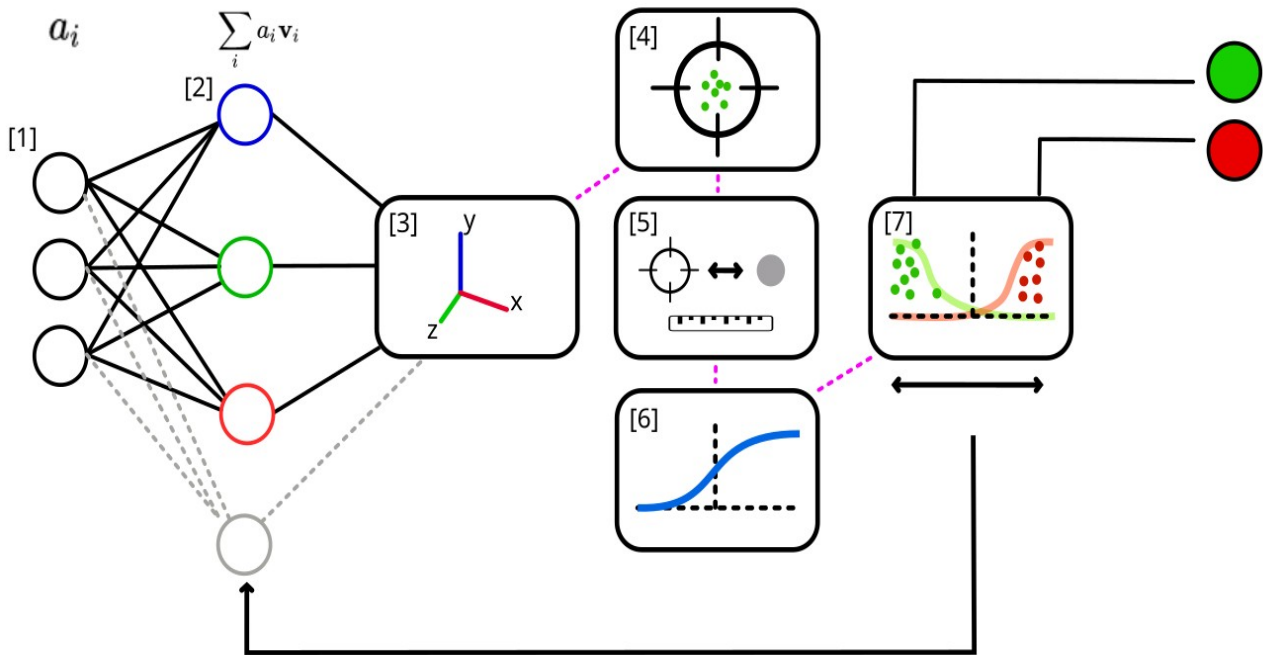
The algorithm

Fig. 7: Sketch of the algorithm used for classification training

In each pass one class is set as target. All other classes are equally
non-targets. Aim of a pass is to separate targets from non targets based
on characteristic relations (weighting) of variables.

Variables are the fundamental data [1]. They are multiplied initially by
random vectors in a linear combination [2]. Independently, multiple
linear combinations are calculated (see 'numDims') [2]. The results are
regarded as multi-dimensional points for sample [3]. The target class's
average/median is set to 'center 0|0|0...' [4]. The distance to the
'center' is determined for all samples [5]. All distances are weighted by
a sigmoidal function [6] to produce scores between 0 (close to target)
and 1 (far away from target). A split between target/non-target is
determined for classification [7]. Back-propagation via adjustment of
vectors [7]>[2] is used to increase separation (>bonus).
If no further improvement of separation is achieved for steps (see
'noBetterStop') the current pass is finished. The respective vectors and
results are one model. Next class is set as target and a new pass is
initiates until all classes are passed through (cycle). Depending on the
'numEnsemble' value cycles is are repeated to finally form the model
ensemble.