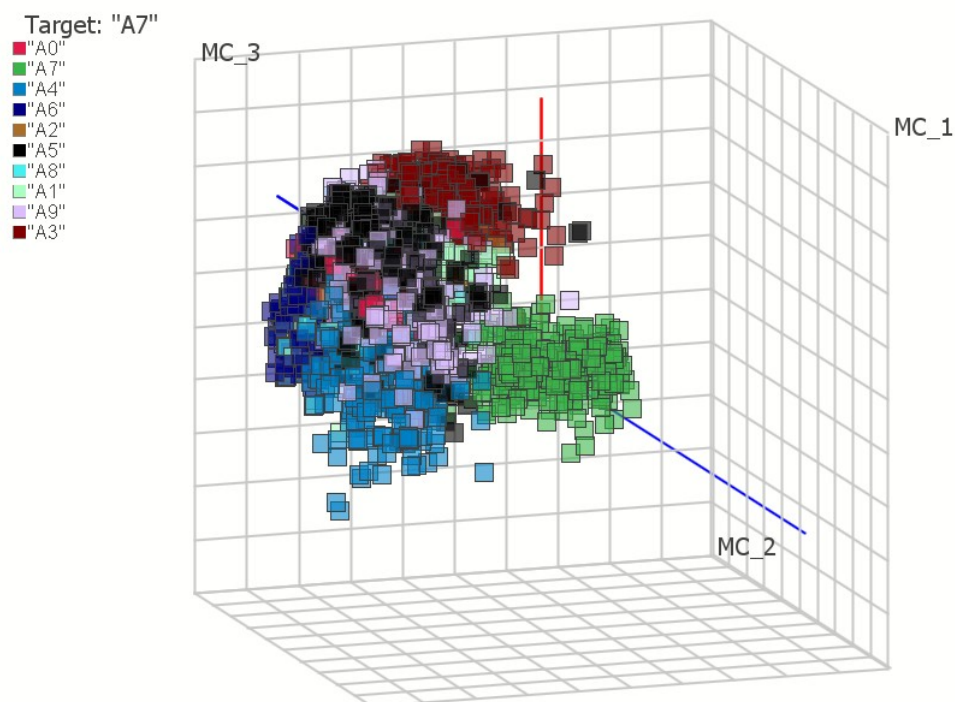**Solver Tutorial**
v.01 DRAFT
26.05.2024

Contact: dsanders@gmx.net

Classification Model Generation and Application

using Solver

Solver version 0.5 rev. 144 updated 26.05.2024

download here: https://github.com/dsandersGit/GIT_Solver

Table of Contents

©2024 Daniel Sanders

When to use

- You have a data set with two or more sample-classes and seek for an automated impartial process to create a classification model for future class estimation of new samples

  - *load and train your data set > save the ensemble > apply ensemble on new data*

- You want to discover which features predominantly classify samples into classes

  - *load and train your data set > check 'importance' in the 'Trends Tab'*

- You want to test if samples-classes can be classified by given features at all

  - *load and train your data set > check the validation and confusion matrix*

- *You want to learn or explain how basic machine learning works*

  - *traceable algorithm and results for learning and education*

  - *easy simulation of data set in spread sheet for testing and demonstration*

Algorithm

The classification model creation can be seen as a data fitting that increases the distance of selected 'target' sample-class from all other sample-classes in a virtual space of multiple dimensions. A set of eigenvectors, multiplied to respective sample feature values, will determine the position of each sample in space (see Visual Math of Spatial Isolation / Linear Combination). These eigenvectors will then be optimized in order to separate samples by their classes. Sample's distance, normalized by a sigmoidal function, represent the 'score' of a sample. Scores then allows the estimation of a class affiliation. A scheme can be found here: Scheme Algorithm.

The calculations of an optimized model are similar to a LDA.

Beneficially for dealing with multi-class data, one model is generated for each class. The models then are used to estimate the probability of an unknown sample's affiliation to the respective class based on their scores.

The training of all classes typically is performed multiple times ('cycles') in order to overcome random local minimum relaxation issues.

The collection of models is combined in an ensemble, that estimates the class affiliation of new samples.

Data Set Syntax and Importing

The data set is a matrix of samples with sample-classes and features.

Features: A set of numerical values describing a sample property, e.g. a measurement result or scalable property. The feature values are used for classification model optimization.

Sample-class: Each samples must be attributed to a sample-class. Sample-classes should consist of multiple samples. Sample-classes are defined by a respective text string. All samples that carry this string are assigned to the respective sample-class. During model building the optimization for classification tries to assign samples to their respective classes. During model application a sample class is estimated for each sample.

Sample: One individual set of features, belongs to one sample-class. A sample is defined by one line in the matrix.

Figure 1 exemplarily describes a data set, compiled of two classes ('good','bad) and four features (1...4).

	A	B	C	D	E
1	CLASS	Feature 1	Feature 2	Feature 3	Feature 4
2	GOOD	79.5	89.3	63.1	7.1
3	GOOD	82.9	3.6	81.6	58.8
4	GOOD	14.9	6.5	19.6	16.4
5	GOOD	12.0	56.1	70.6	96.3
6	GOOD	70.1	43.9	72.6	43.4
7	GOOD	42.8	13.7	52.1	62.1
8	BAD	8.5	59.1	2.1	66.1
9	BAD	58.7	4.1	87.1	49.9
10	BAD	2.7	99.0	67.7	99.0
11	BAD	70.5	40.3	44.6	73.5
12	BAD	33.5	84.9	71.8	72.3
13	BAD	65.5	43.8	92.0	44.5

Figure 1: Exemplary Data Set in spread sheet

Reasonable data sets consist of:
Features: 2 – 100
Samples: 20 – 10000
Classes: > 1
Samples per Class: > 10

In general, models created by large sample numbers and small feature numbers are more generalizable. Even fully random feature values of a small sample count with large feature number will result in a pretended good training accuracy ('curse of dimensionality'). Always check validation.

Import from Clipboard

Data can be imported by copying from spread sheet programs (separator is <TAB>) and imported into solver using <CTRL> + V or by <menubar><File><Import data from Clipboard>. The class column must not be at column #1. Samples must not be sorted by sample-class.

Import from CSV File

Data can be imported from CSV files (comma separated) by <menubar><File><Load Dataset>. See figure 2 for syntax. The class column must not be at column #1. Samples must not be sorted by sample-class.

```
CLASS,Feature 1,Feature 2,Feature 3,Feature 4
GOOD,79.5,89.3,63.1,7.1
GOOD,82.9,3.6,81.6,58.8
GOOD,14.9,6.5,19.6,16.4
GOOD,12.0,56.1,70.6,96.3
GOOD,70.1,43.9,72.6,43.4
GOOD,42.8,13.7,52.1,62.1
BAD,8.5,59.1,2.1,66.1
BAD,58.7,4.1,87.1,49.9
BAD,2.7,99.0,67.7,99.0
BAD,70.5,40.3,44.6,73.5
BAD,33.5,84.9,71.8,72.3
BAD,65.5,43.8,92.0,44.5
```

Figure 2: Exemplary Data Set in CSV Syntax

Import from DAT File

Specific file format, not recommended for general use.

Import during startup

Solver can be configured during start-up to load a defined data set file directly.

After importing

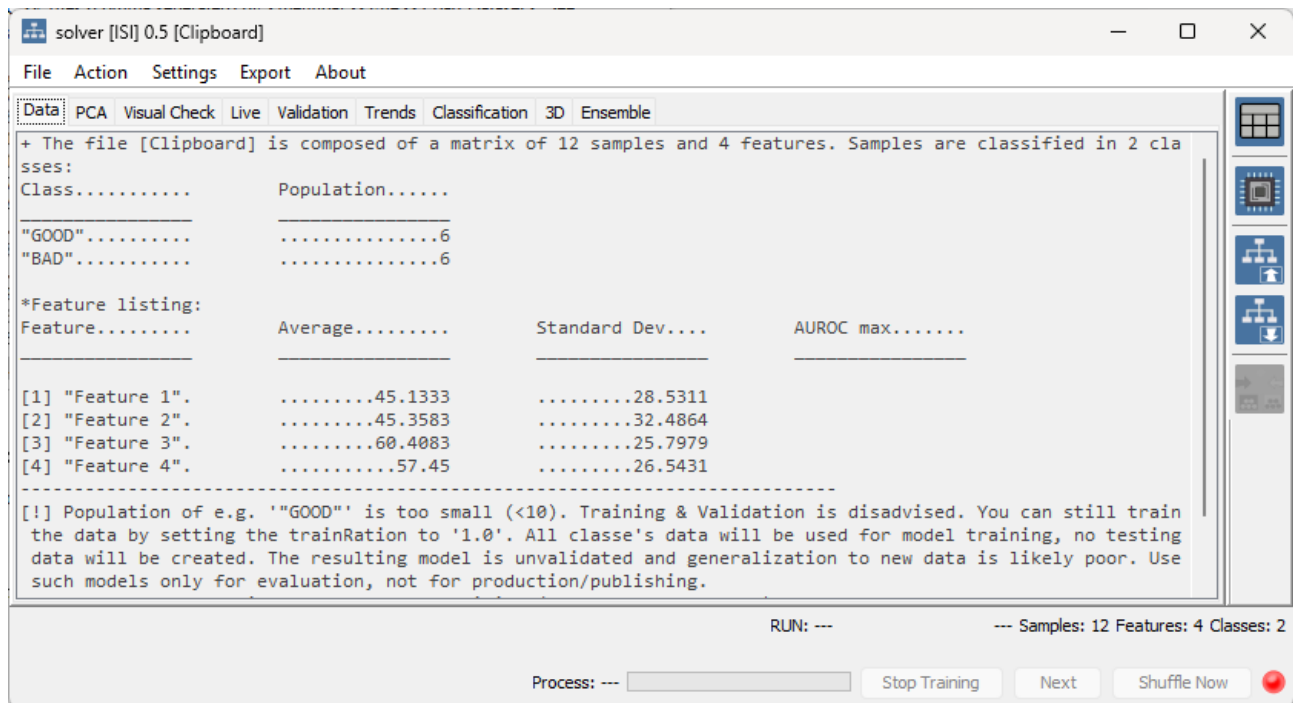The data tab will show a summary of the imported data:

Figure 3: Solver: Data Tab with data set summary

Number of classes, samples, features, class populations (samples per class) are listed. Furthermore, the average and the standard deviation for the feature values are presented.

Note, for small sample per class number a training is not advised.

Running a classification model training

Preparation

Visual Check

Before running a training process it is advised to visually check the data in the 'Visual Check' Tab.

Walking through the plots of feature values versus sequence (the original order of the samples) is crucial in order to ensure validity of the data. Any sort of gradient or step that is not related to class assignment asks for re-checking of potential data wrongness. The sensor that records a feature might have a gradient, or features recorded with different sensors (> steps) might significantly alter the outcome of the machine learning: *'bullshit in leads to bullshit out'*. It might be beneficial to alter the sample sorting time dependant to observe gradients or steps. Alternatively, an extra feature with the label 'timeindex[label]' with time data can be added to the data set. This feature is not used as classification feature but can be used here as x-axis.
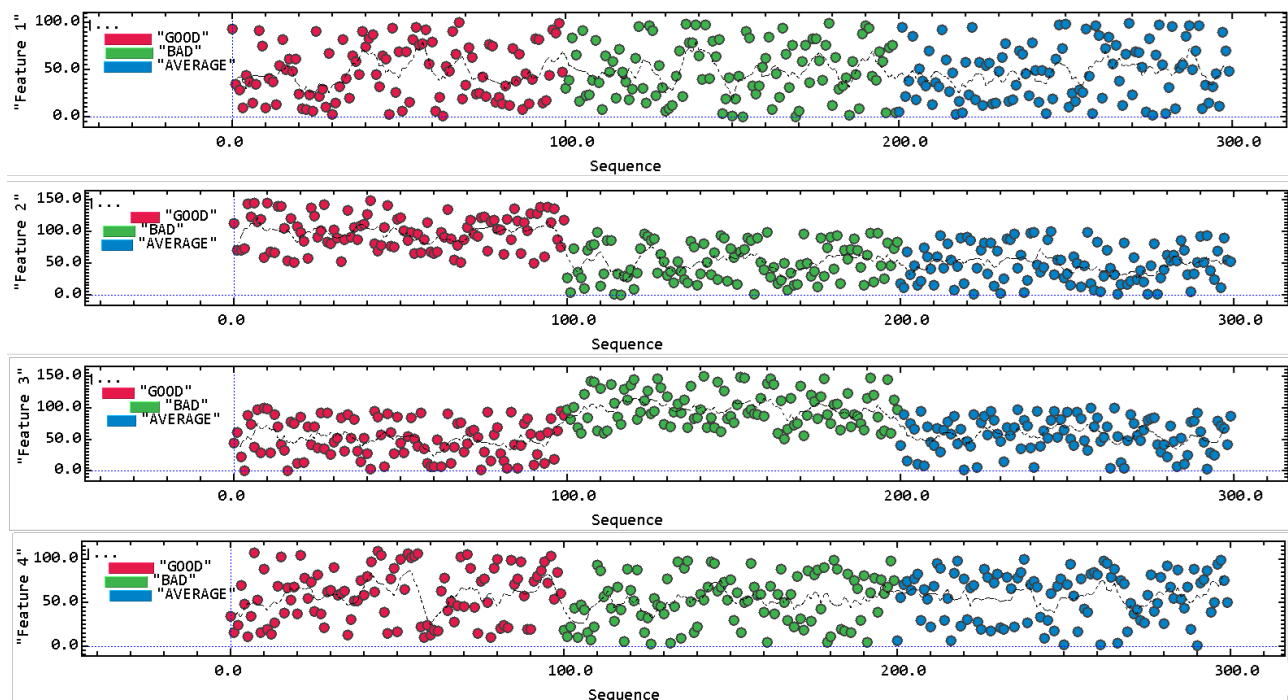
To walk-though use the mouse wheel of the arrow buttons.

Figure 4: Visual Check of the Mock-up data set

The mock-up data do not show gradient and only exhibit the intended steps (e.g. feature 2 is elevated for 'good', feature 3 for 'bad').

When discovering suspicious features or for model development respective features can be de-activated for the model building by double-clicking on the feature vs sequence plot or by the respective checkbox. The feature's vector will be fixed to 0 in the model process.

Options
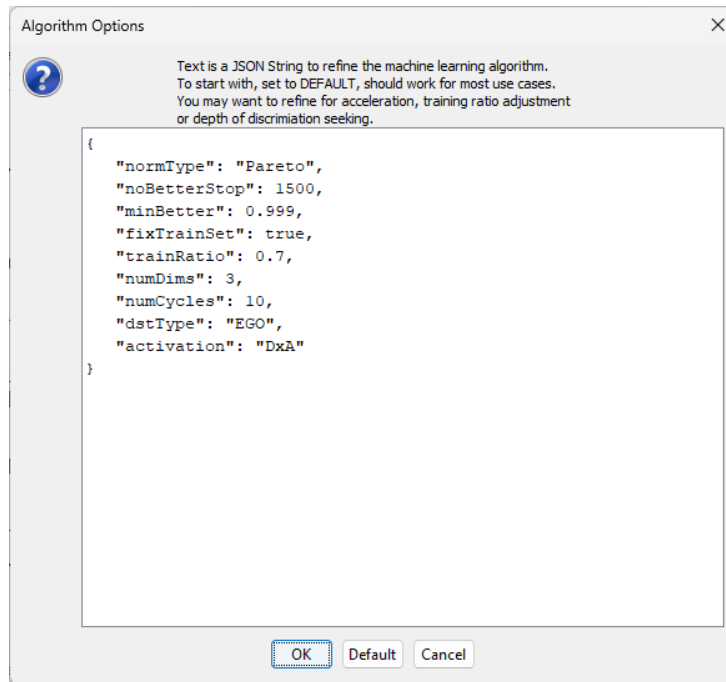
Option dialogue is found in the <menubar><Settings><Algorithm Options>

These option configure the machine learning algorithm and my need adjustment in special use cases. In general, a set of default options is advised as a starting set. Options can be reset by pressing the <default> button.

Normtype
Feature values are normalized in order to weight them equally. Two normalizations are possible:
'Pareto' – Pareto scaling
'MaxMinNorm' – Set smallest feature value to -500, highest to 500

noBetterStop
The algorithm determines that optimization is finished when the class separation is not improving. If so, further 'noBetterStop' times will be tried.

☞ Increasing this value might improve classification results, but will extend the process time. For large sample sets reduce number.

MinBetter
The algorithm determines that optimization is finished when the class separation is not improving. 'minBetter' is the factor that controls the minimal accepted improvement.

fixedTrainSet
During the learning process only a ratio of samples per class will be used for training, the rest is used for validation. See trainRatio. The random selection of training to validation normally is fixed (value: true) but can be set to new random selection for each class (value: false).

dstType
Selected the algorithms mechanism for determination of a training sample importance. Options are 'EGO' and 'GROUP'.

activation
Selects the activation equation used. Options are 'DxA' (Distance times accuracy), 'D' (distance only) and 'A' (accuracy only).

trainRatio
See fixedTrainSet. Sets the training to validation ratio. 1.0 omits validation data, default 0.7 will use 70% of the samples in each class for training and 30% for validation.

numDims
Classification model used multi-dimensional separation. The number of dimensions can be set. Minimum 1 only used a single linear separation. Default is 3 dimensions. Note, the 3D plot is only accessible when numDims is 3.

☞ Set to 1 to study feature importance (classification power might be reduced, though)

numCyles
The training process can run multiple cycles of individual classification trainings. For each class one model is created (respective class is target) per cycle. So, for three classes and one cycle three models will be generated in order to estimated the class of an unknown new sample. Multiple cycles can be passed through to build a larger model set. In general larger model set estimates are more accurate and robust. Training time will be higher for multiple cycles.

☞ Increasing this value might improve classification results, but will extend the process time.

Staring a Training

<menubar><Actions><Train> or [icon] in the right hand side icon bar will initiate a training process. The process can be monitored in different tabs:
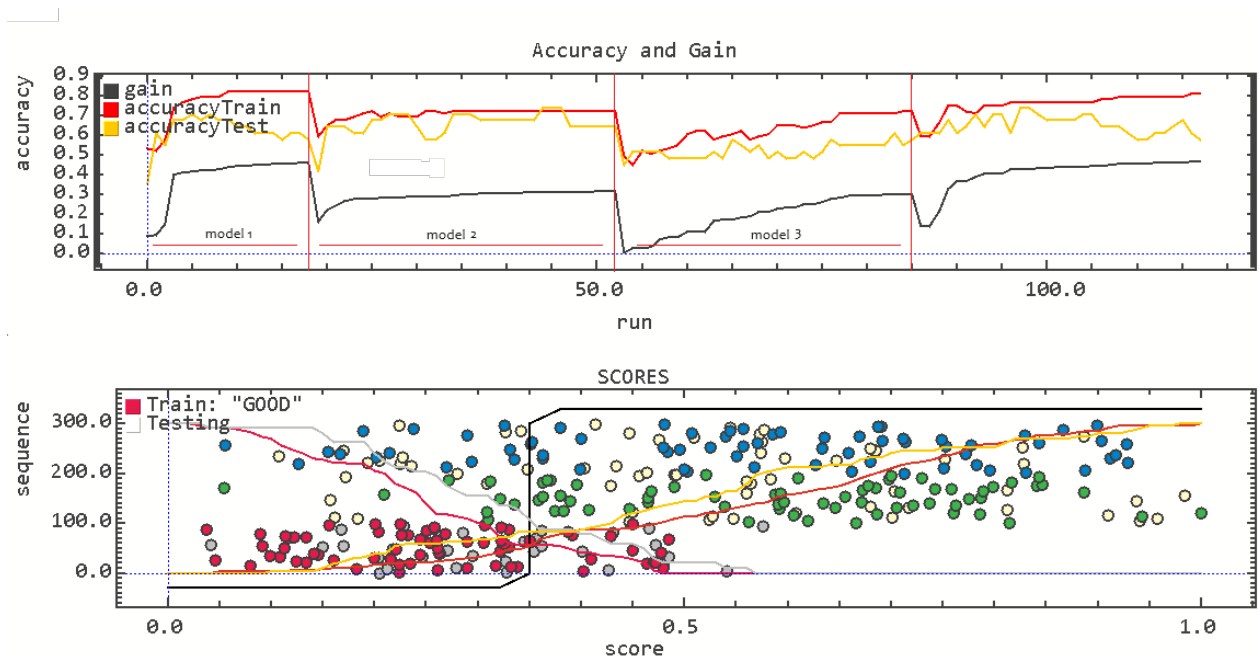
Live Tab

Figure 5: 'Live' View

Default view during model training is the 'live' view.

Model building iteratively selects one sample-class as target and tries to discriminate all other samples, independent of their class, by spatially increasing their distance from the average position of the targets in a N-dimension space ('scores' are the distances after sigmoidal normalization, see Visual Math of Spatial Isolation / Linear Combination). The scores are live plotted as x-axis on the bottom plot. Y-axis is the sample's sequence (order during import). Optimization can be followed by monitoring samples of the current target class (here: 'GOOD' in red) to the left and all other samples to higher scores. Sample colors are automatically selected by the class order during import. Colored filled dots samples are training samples, while gray / light-yellow filled dots are validation samples.

The class colored line indicates the integration of target-class samples (integration 1 → 0), while the red line (0 → 1) integrates non-target samples. A score-split is determined where the two integrals meet. Gray / light-yellow curves indicate the integral of the validation data.

The top plot shows the progress of the trainings process. During each model-building the gain (internal optimization function) and the accuracy of training and validation estimates are plotted on the overall run of the training. Well aligned training and validation accuracies indicate a good

generalizability of the model. When the training of one target is finalized, the values for gain and accuracy drop when a new target is trained.

The gain equation is selectable in the options, see Options dstType and Options activation.

Training can be stopped (! models will be cleared) by the button in the status bar. The 'Next' button will terminates the current model and progresses with the next one. 'Shuffle now' resets random initial vectors.

Validation Tab

The 'validation' tab compiles the sensitivity, specificity and accuracy of both training and validation class estimated. Each row will show the respective values together with the target.

Data	PCA	Visual Check	Live	Validation	Trends	Classification	3D	Ensemble							
run	type	Target	TP_Train	FP_Train	TN_Train	FN_Train	Sensitivit...	Specificit...	TP_Valida...	FP_Valida...	TN_Valida...	FN_Valida...	Sensitivit...	Specificit...	Accuracy...
000001	DxA	"GOOD"	53	31	107	16	0.7681	0.7754	23	16	46	8	0.7419	0.7419	0.7419
000002	DxA	"BAD"	51	36	102	18	0.7391	0.7391	19	18	44	12	0.6129	0.7097	0.6774
000003	DxA	"AVERAGE"	45	46	92	24	0.6522	0.6667	17	15	47	14	0.5484	0.7581	0.6882
000004	DxA	"GOOD"	54	31	107	15	0.7826	0.7754	24	15	47	7	0.7742	0.7581	0.7634
000005	DxA	"BAD"	51	38	100	18	0.7391	0.7246	22	23	39	9	0.7097	0.629	0.6559
000006	DxA	"AVERAGE"	44	45	93	25	0.6377	0.6739	20	18	44	11	0.6452	0.7097	0.6882
000007	DxA	"GOOD"	53	34	104	16	0.7681	0.7536	23	16	46	8	0.7419	0.7419	0.7419
000008	DxA	"BAD"	49	41	97	20	0.7101	0.7029	21	22	40	10	0.6774	0.6452	0.6559

Figure 6: 'Validation' Tab

TP – true positive, FP false positive, TN – true negative, FN – false negative

Further explanation of the validation data can be found here:

https://en.wikipedia.org/wiki/Sensitivity_and_specificity
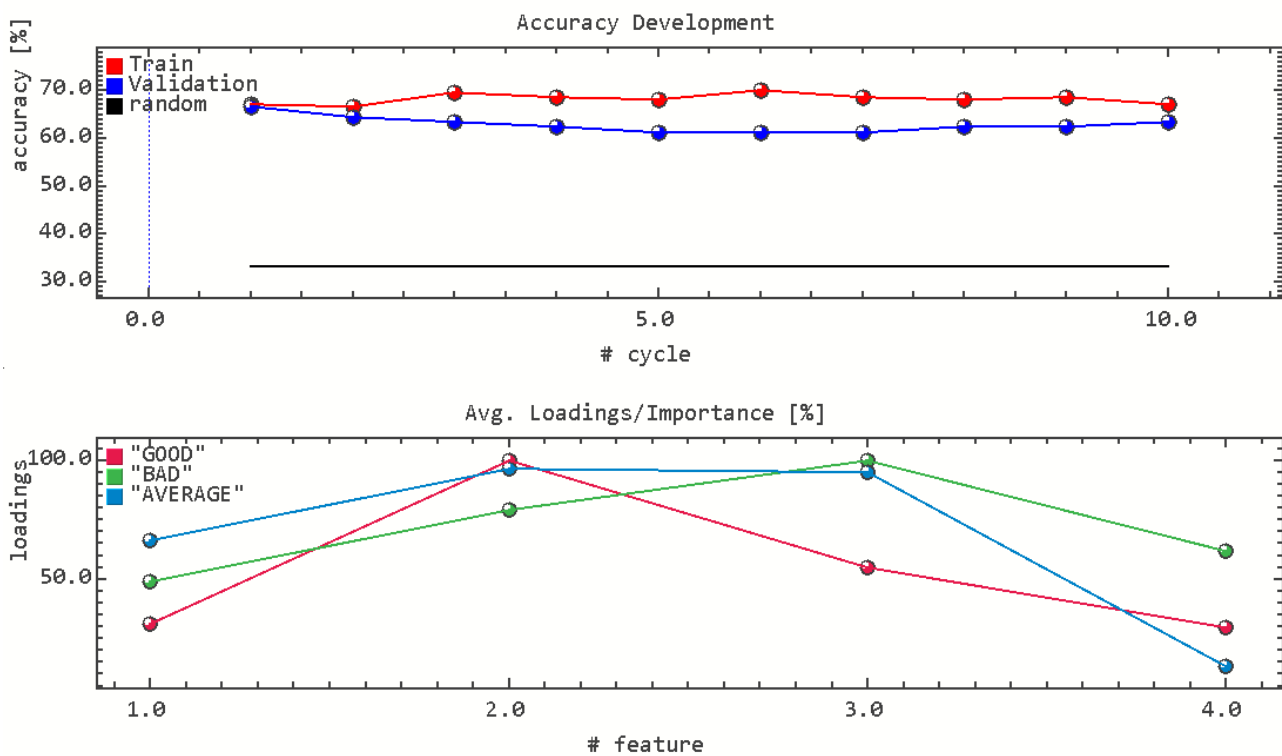
Trends Tab

Figure 7: 'Trends' Tab

During the model building process the accuracies of training and validation data are plotted on the cycle course (top plot). Furthermore, the statistical random classification is pictured for comparison. The generalization quality of the model ensemble can be judged by the spread between training and validation accuracies.

The bottom plot depicts the importance of the loadings for model building for each sample-class. The importance is calculated by the average of the maximal feature vector length. Applied on the mock-up data set (Mock-up Data Set) we learn that features 2 and 3 bear the highest importance.

Increasing the cycle count and reducing the number of dimensions (Options) increases the explanatory power of the importance.

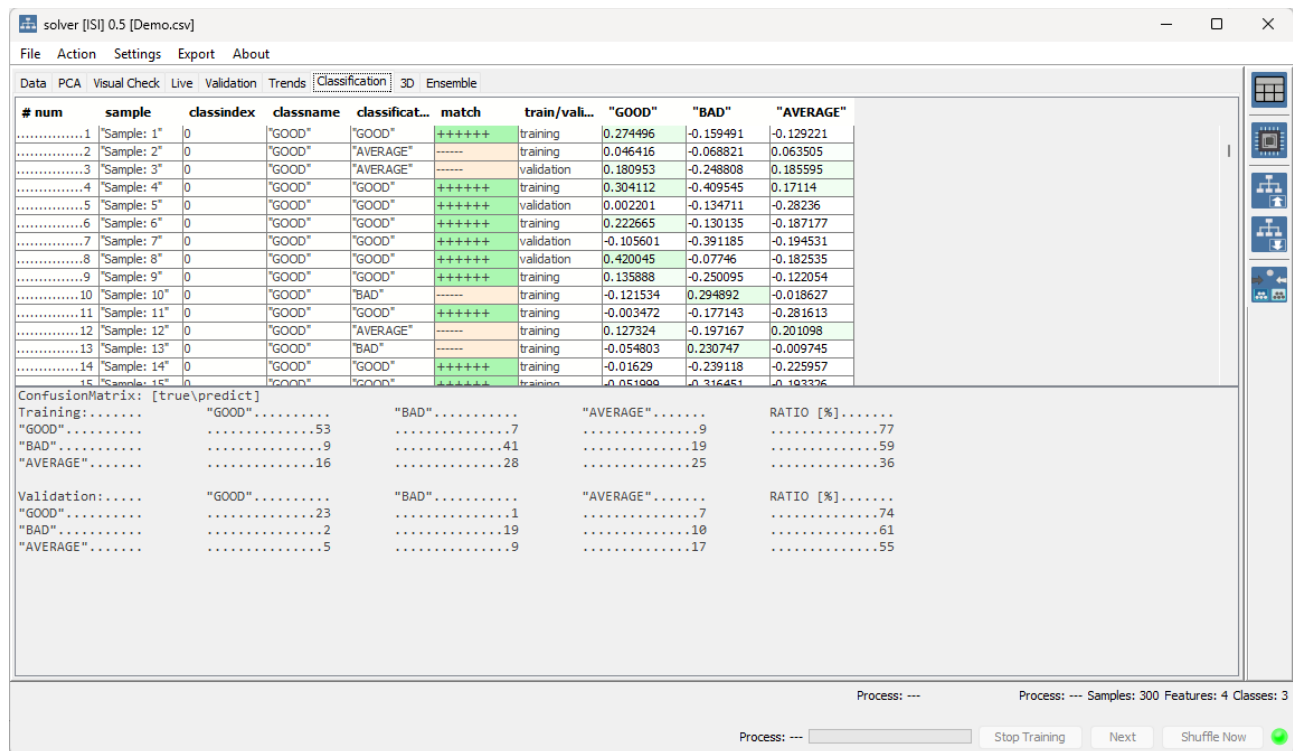Plots are updated after each cycle.

Classification Tab

Figure 8: 'Classification' Tab

After each cycle the current model ensemble is used to classify all data. Usage of sample as training and validation is indicated. The classification scores indicate the probability of a sample to be member of the respective class. Depending on the Options setting scores 0 – 1 or -1 to 1 are calculated.

Classification is summed up in a confusion matrix. Rugged model ensembles will exhibits similar accuracy ratios.

Classification is updated after each cycle.
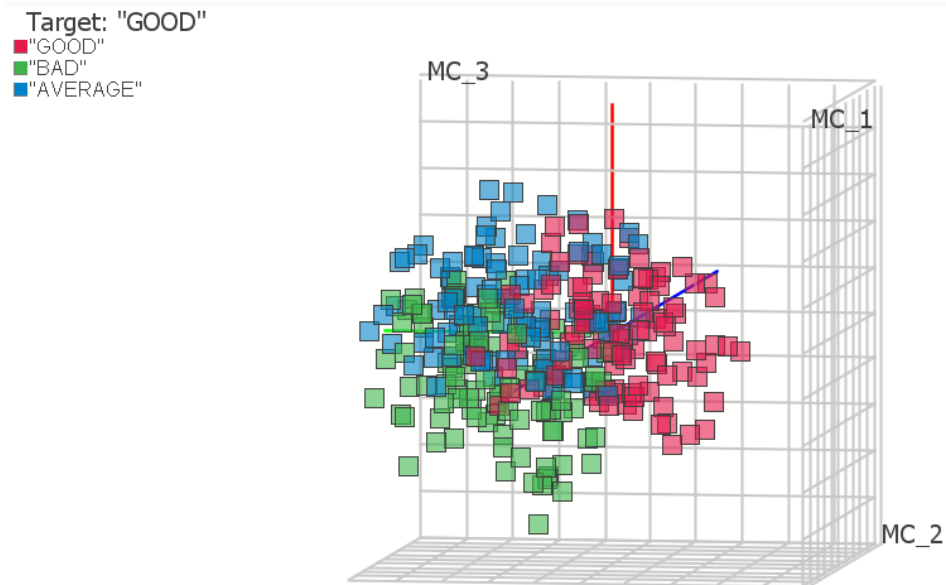
3D Tab

Figure 9: '3D' Tab

'3D' is a pseudo three dimensional visualization of the current model adjustment process.

The centering of the target class and the distances leading to the scores in the Live Tab can be followed visually.

Accessible only when number of dimensions is three.

Training is finished

Training process is displayed in the progress bar. After finishing of the training a new model ensemble is accessible in the 'Ensemble' tab.
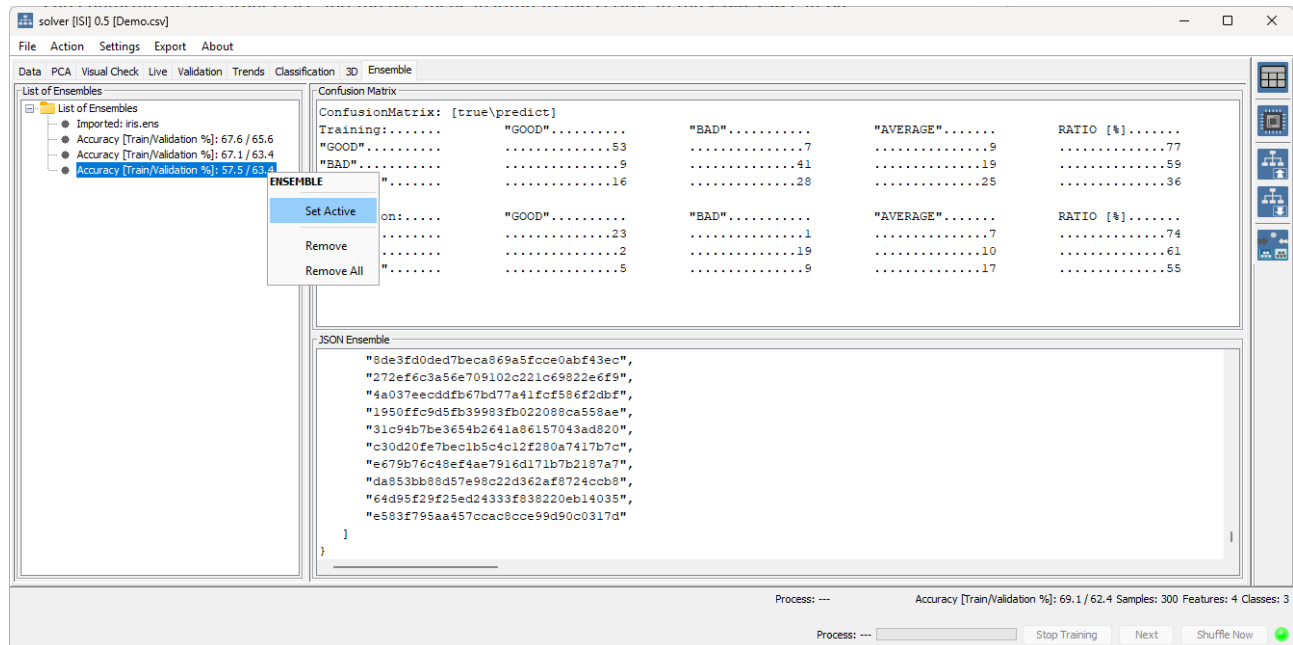

Figure 10: 'Ensemble' Tab

Selecting an ensemble (right mouse click > Set active) will set the underlying ensemble as active. A new classification is based on the active ensemble.

The active ensemble's confusion matrix and the full ensemble data is displayed.

When saving the active ensemble is saved as text file in JSON format as displayed.

The last saved ensemble is stored and will be loaded when restarting.

Always critically question your models, especially if the separation power is high. Check the features for gradients and steps, see Visual Check. Check Options. You may verify by removing some initial classified samples and using them as testing data set.

Applying a Classification Ensemble

Loading an Ensemble

Saved classification model ensembles can be loaded by <menubar><File><Load Ensemble> or via the right-hand side icon bar. The loaded ensemble is set to active.

Importing a Data Set for Classification

Data can be imported as described for training: Data Set Syntax and Importing

Note, the feature labels must be exactly equal to the labels used during training of the ensemble.

Note, *Solver* can be configured during start-up to load a defined data set file directly.

Apply Classification

Select <menubar><Actions><Classify> or press the classification button in the right-hand side icon bar. The Classification Tab will be selected where the samples are classified using the underlying ensemble.

The classification can be exported by <menubar><Export><Classification>. The classification table is now accessible in the clipboard for pasting into spread sheet programs (elements are separated by <TAB>).

Note, during classification the data's sample-classed are ignored. Therefore a confusion matrix can not be calculated.

Mock-up Data Set

Throughout this manual a mock-up data set is used for demonstration:

|	A CLASS	B Feature 1	C Feature 2	D Feature 3	E Feature 4
1	**CLASS**	**Feature 1**	**Feature 2**	**Feature 3**	**Feature 4**
2	*GOOD*	17.6	55.5	43.9	104.3
3	*GOOD*	26.0	58.4	61.3	75.6
102	*BAD*	18.5	38.8	122.5	12.2
103	*BAD*	63.4	86.0	61.1	38.2
206	AVERAGE	65.7	23.2	25.0	76.1
207	AVERAGE	78.3	24.7	13.3	56.9

RANDOM [0-100]   RANDOM [0-100] + 50   RANDOM [0-100] + 10

Figure 11: Mock-Up Data Set

Classes are 'good', 'bad' and 'average'. Each 100 samples per class were created in a spread sheet tool. Features are generally a random number [0-100]. Feature 2 values (class 'good'), Feature 3 values (class 'bad') are elevated by 50. Feature 4 values for 'good' is elevated by +10. See figure 4.

The elevation of the feature values for distinct classes should allow a basic, yet unperfect, separation.

User Interface
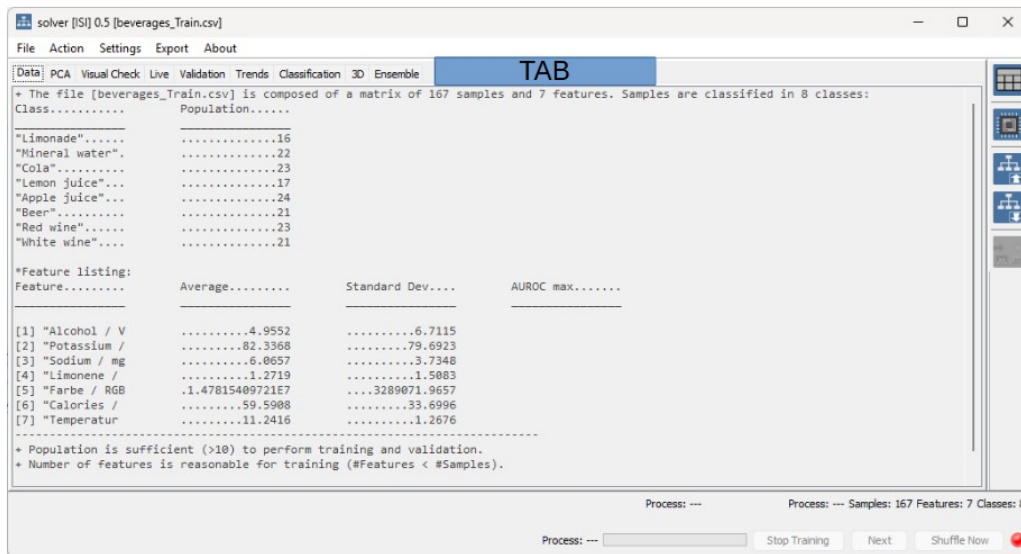
Menubar

File – import and save data sets, load and save ensembles

Actions – Start and Stop Training
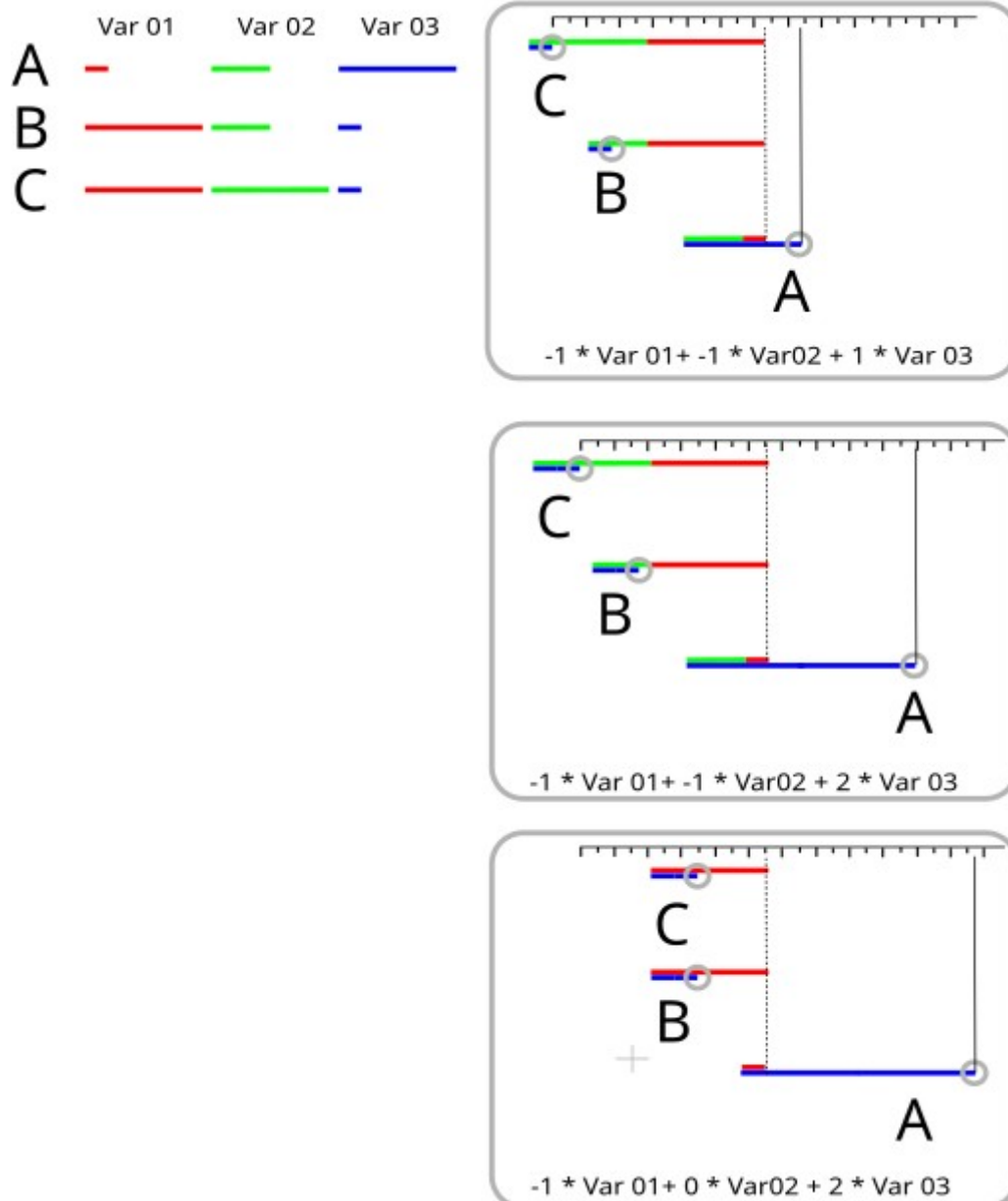
Settings – Algorithm options, further settings

Export – data to clipboard, training scores, classification, validation

About

Appendix

Visual Math of Spatial Isolation / Linear Combination

-1 * Var 01+ -1 * Var02 + 1 * Var 03

-1 * Var 01+ -1 * Var02 + 2 * Var 03

-1 * Var 01+ 0 * Var02 + 2 * Var 03

Samples A, B and C are described by the feature values Var_1, Var_2 and Var_3, represented by lines of different lengths. The plots show the one dimensional result of a linear combination of the features and exemplary eigenvectors (first: -1, -1, 1). Changing the eigenvectors results in a change of the linear combination. Note that the third example results in matching positions of B and C, although feature values are different. Solver used such a calculation to separate sample-classes. The vectors are changed and the result is compared automatically, If the separation increases the change is confirmed, otherwise it is undone.

Scheme Algorithm

As used for one target / model