PyCR: A hybrid backwards elimination/forward selection wrapper algorithm to select informative variables that maximise cluster resolution

Wenwen Li
Michael Sorochan Armstrong
A. Paulina de la Mata
James James Harynuk

March 2022

1 Introduction

PyCR (Cluster Resolution in Python) is a variable selection algorithm that examines the effect of each variable in a dataset within the context of highly-ranked variables by measuring the maximum confidence interval over which two or more clusters are separable within principal component space. This measurement is defined as cluster resolution (ξ for binary classes, or Ξ for 3 or more classes)[2].

The algorithm works by incorporating highly-ranked variables and individually removing them from the list of variables being considered. Variables whose absence improves the cluster resolution are discarded for an entire iteration of the algorithm, during this backwards elimination step. Once the last highly-ranked variable, as defined by a statistically-informed threshold, has been considered, poorly ranked variables are subsequently added to the variable list and the resultant cluster resolution is measured. As before, variables whose inclusion improves the cluster resolution of the model are included for a particular iteration of the algorithm[4]. This forward selection step proceeds until a statistically informed threshold is reached and no further variables are selected for inclusion.

The algorithm iterates several times, after which time variables that exceed a particular survival rate are included for selection in the final model. PyCR also automatically provides critical statistics and graphics illustrating the performance of an SVM classifier before, during, and after feature selection.

2 Installation

You can download the necessary code, or make a clone from the GitHub repository: `https://github.com/Rikyyyyyyyyy/PyCR`. Unzip the files to a convenient directory. All input and output data will take place in this directory.

In order for PyCR to work, you will need to have Python3 installed on your computer, and the PyCR module downloaded from the GitHub repository. Once Python3 is installed, and you are

able to call it directly from the terminal (for Windows systems, Python3 must also be added to the path) navigate to the repository directory in the terminal and execute the following command to install the necessary modules:

```
pip3 install -r requirements.txt
```

The necessary modules can also be installed manually, by following the list of required packages in the requirements.txt document.

3 Inputs

There is a example dataset that can be used that is downloaded as part of the PyCR package. It is recommended that you test the algorithm on this data before uploading your own data.

Required declarations:

`python3`

`PyCR.py`

1. External Validation `True/False`
2. Split ratio (External Validation) `0-1`
3. Multiple ROCs per figure (For 3 or more classes) `True/False`
4. Total Useful Peak Area (TUPA) normalisation `classtupa/tupa/notupa`
5. Input data in MetaboAnalyst format? `True/False`
6. If MetaboAnalyst data file, file name. If none use: `None`
7. Data file directory/name `./filename`
8. File containing class information `./filename`
9. File containing variable names `./filename`
10. Desired scaling `Autoscale/SVN`
11. Iterations `0-200`
12. Variable survival rate `0-1`

Example terminal command:

```
python3 PyCR.py True 0.5 False classTupa False None
    Input/data_algae.csv Input/class_algae_string.csv
        Input/S_name.csv Input/v_name.csv AutoScale 1 0.85
```

Each file should be in `.csv` format, with explicit "," delimiters.

External validation is optional, but highly recommended. The algorithm will take care of all critical statistics and validation necessary for an untargeted metabolomics, or similar experiment. The split ratio defines what percentage of the data will be used for external validation.

For multiple classes, the user has the option of exporting all Receiver-Operator Characteristics for pair-wise classification performances as one figure, or several figures. For higher class numbers (> 3) this may make the figures difficult to read.

Total useful peak area (TUPA) is a very useful metric for sample-wise normalisation in untargeted metabolomics studies. TUPA has been demonstrated on urine metabolomics data, and class-based TUPA [3] has been deployed for comparison of samples whose biological make-up is poorly comparable. Please note that because cTUPA [1] operates on *a-priori* class information, it is not possible to use for predictive models. Instead, it is a useful tool for metabolomic profiling.

Data can either be input using the MetaboAnalyst format `metaboanalyst.ca`, or using explicit ".csv" files containing the class, data, and variable names.

Auto-scaling (typical for chromatographic peak table data), Standard Normal Variate (SVN) scaling (more typical for spectroscopic data) can performed on the data. Following SVN, all data is additionally mean-centred.

Higher numbers of iterations are always more desirable, but will take a longer time to execute. The survival rate is calculated out of the total number of iterations a variable is selected.

4 Outputs

The goal of this algorithm is to perform all of the necessary plots and analyses necessary for the validation of an untargeted metabolomics analysis. All necessary information can be found in the `output` folder, containing the following files and directories:

The animation folder contains all images used in the construction of the file `animation.gif`, illustrating the action of the algorithm selecting features during the first iteration.

`rocExternal` is a directory containing the ROC curves for the external validation set.

`rocIterations` is a directory containing the ROC curves for the internal validation within the training set.

`rocTrainFS` contains ROC curves for the training set using the selected features.

`rocTrainNoFS` contains the ROC curves for the training set using all features. This is to determine if the feature selection has improved the performance of the linear Support Vector Machine (SVM) classifier on the training data.

`rocValiFS` contains the ROC curves for the validation, or test set using the selected features.

`rocValiNoFS` contains the ROC curves for the validation set using all features. Again, this is to help determine if the feature selection step has helped the predictive performance of the model.

`auc_table_class_X.csv` contains the area under the curve value for class X against all other classes for each iteration in the training set.

`ClassTupaAllSample.csv` contains the TUPA normalisation values.

`confusion_matrix.csv` contains the information on the predicted and true classes for the external set using the selected features.

`confusion_matrix_no_FS.csv` contains the information on the predicted and true classes for the external set using all of the features. This is to determine if the feature selection step has helped the classification performance.

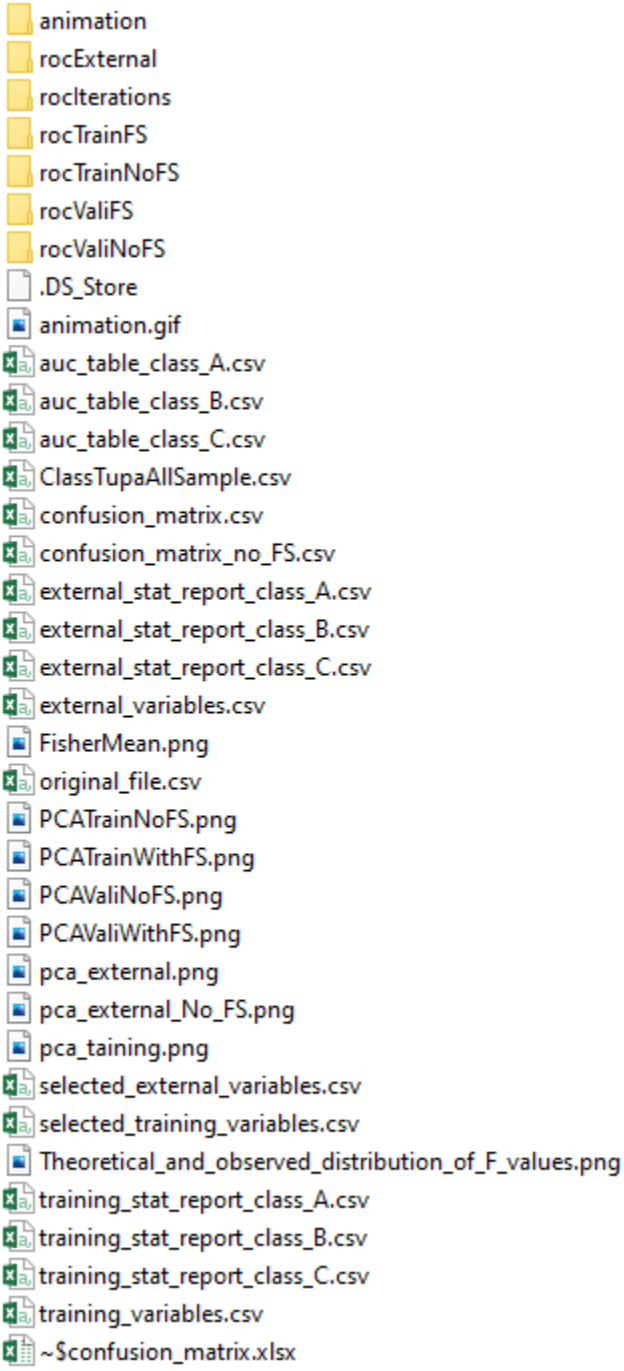

Figure 1: Caption

`external_stat_report_class_X` contains the critical statistics for classification performance for class X against all other classes.

`external_variables.csv` contains the external validation set with the all variables in MetaboAnalyst format.

`FisherMean.png` displays a graphic summary of the determination of the start and stop numbers.

`original_file.csv` contains the original data in MetaboAnalyst format.

`pca_external.png` contains the principal component scores of the training and validation sets using the selected features.

`pca_external_No_FS.png` contains the principal component scores of the training and validation sets using all of the features. Useful for determining whether or not the feature selection step has helped the model.

`PCATrainNoFS.png` is a plot of the principal component scores without feature selection only using samples in the training set.

`PCATrainWithFS.png` is a plot of the principal component scores with feature selection using only samples in the training set.

`PCAValiNoFS.png` is a plot of the principal component scores of the validation set with all features.

`PCAValiWithFS.png` is a plot of the principal component scores of the validation set with the selected features.

`selected_external_variables.csv` contains the external validation dataset in MetaboAnalyst format with the selected variables.

`selected_training_variables.csv` contains the training dataset in MetaboAnalyst format with the selected variables.

`Theoretical_and_observed_distribution_of_F_values.png` contains the observed distribution of F values overlaid on a plot of the theoretical distribution with the corresponding degrees of freedom.

`training_stat_report_class_X.csv` contains the classification statistics for the training set of class X versus all other classes.

`training_variables.csv` contains the training set with all variables in MetaboAnalyst format.

The following papers are relevant to the FS-CR algorithm, and normalisation to the total useful peak area (TUPA):

References

[1] Michael D Sorochan Armstrong et al. "Global metabolome analysis of Dunaliella tertiolecta, Phaeobacter italicus R11 Co-cultures using thermal desorption-Comprehensive two-dimensional gas chromatography-Time-of-flight mass spectrometry (TD-GC× GC-TOFMS)". In: *Phytochemistry* 195 (2022), p. 113052.

[2] Michael Sorochan Armstrong, A Paulina de la Mata, and James J Harynuk. "An efficient and accurate numerical determination of the cluster resolution metric in two dimensions". In: *Journal of Chemometrics* 35.7-8 (2021), e3346.

[3] Seo Lin Nam et al. "Towards standardization of data normalization strategies to improve urinary metabolomics studies by gc× gc-tofms". In: *Metabolites* 10.9 (2020), p. 376.

[4] Nikolai A Sinkov and James J Harynuk. “Cluster resolution: A metric for automated, objective and optimized feature selection in chemometric modeling”. In: *Talanta* 83.4 (2011), pp. 1079–1087.