Author: RNDr. Eva Janoušová
Institute of Biostatistics and Analyses, Masaryk University, Brno

# Penalised Reduction & Classification Toolbox

## Created by Eva Janoušová

Penalised Reduction & Classification Toolbox provides algorithms for reduction and classification of various types of data, such as genetic data, two-dimensional (2-D) face image data or three-dimensional (3-D) brain image data. The 2-D and 3-D data of each subject or object has to be transformed into 1-D vector and stored as rows in a data matrix prior to entering the toolbox.

The algorithms were implemented as functions in MATLAB® environment. The list of the functions including purpose of the function, a syntax example and superordinate functions is given in Tab. 1. The input and output data and variables of the functions are listed in Tab. 2. A hierarchy of the functions included in the Penalised Reduction & Classification Toolbox is depicted in Fig. 1.

The Penalised Reduction & Classification Toolbox enables reduction of data by selecting most discriminative features using penalised linear discriminant analysis (pLDA) with resampling and penalised linear regression (pLR) with resampling. Moreover, the toolbox allows also for feature selection using t-test and feature extraction using principal component analysis (PCA), which are both commonly used methods and serve as a standard for comparison with results achieved using pLDA and pLR. The reduced data are then classified into two groups using linear discriminant analysis (LDA) or linear support vector machines (SVM). The toolbox can be easily extended to comprise more data reduction and classification methods. An output from the toolbox is a classification performance and identified most discriminative features which can be visualized. The classification performance measures, specifically accuracy, sensitivity, specificity and precision, are calculated based on leave-one-out cross-validation (LOOCV). In LOOCV, every subject/object (a row vector of the input matrix) is chosen as a testing one stepwise and the remaining subjects/objects are used for training the classifier. The toolbox allows performing LOOCV during data reduction and classification (more correct approach) or only during classification. The classification performance of different methods can be compared using the McNemar's test.

Tab. 1: A list of functions included in the Penalised Reduction & Classification Toolbox.

| **Function name** | **Purpose** | **Syntax example** | **Superordinate function(s)** |
|---|---|---|---|
| classif_isPCA_loo.m | Perform classification of 2 groups of subjects/objects based on data reduced using intersubject principal component analysis (isPCA). | [effic,class,gis] = classif_isPCA_loo (X,y,pat,classif,loo,m) | |
| classif_pLDA_loo.m | Perform classification of 2 groups of subjects/objects based on data reduced using penalised linear discriminant analysis (pLDA) with resampling. | [effic,class,gis,nit,fsel, nsel] = classif_pLDA_loo (X,y,pat,classif,loo,thresh,lambda,prob,B) | |
| classif_pLR_loo.m | Perform classification of 2 groups of subjects/objects based on data reduced using penalised linear regression (pLR) with resampling. | [effic,class,gis,fsel, nsel]=classif_pLR_loo (X,y,pat,classif,loo, lambda,prob,B) | |
| classif_Ttest_loo.m | Perform classification of 2 groups of subjects/objects based on data reduced using t-test with threshold alpha. | [effic,class,gis,fsel] = classif_Ttest_loo(X,y, pat,classif,alpha,loo) | |
| classif_Ttest_nsel_loo.m | Perform classification of 2 groups of subjects/objects based on data reduced using selection of features with lowest p-values based on t-test. | [effic,class,gis,fsel] = classif_Ttest_nsel_loo (X,y,pat,classif,nsel, loo) | |
| classif_TtestFDR_loo.m | Perform classification of 2 groups of subjects/objects based on data reduced using t-test with FDR correction. | [effic,class,gis,fsel] = classif_TtestFDR_loo (X,y,pat,classif,alpha, loo) | |
| classifiers.m | Classify reduced data using diagonal LDA or linear SVM classifiers. | [class] = classifiers (Xtest,Xtrain,ytrain, classif) | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo |
| cperformance.m | Estimate classification performance. | [effic] = cperformance (class,gis,pat) | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo |
| isPCA_eig.m | Compute projection matrix of isPCA. | [V,pvar] = isPCA_eig(Xc,m) | classif_isPCA_loo<br>isPCA_reduc |
| isPCA_reduc.m | Reduce data using isPCA. | [Xred,pvar,Xmean,V] = isPCA_reduc(X,m) | classif_isPCA_loo |

| Function name | Purpose | Syntax example | Superordinate function(s) |
|---|---|---|---|
| mcnemar.m | Compute p-value using McNemar’s test for comparison of 2 classifiers. | [p] = mcnemar(class2,gis) | mcnemar.m |
| pLDA.m | Perform penalised linear discriminant analysis. | [v,niters] = pLDA(m1, m2,s,lambda,thresh) | classif_pLDA_loo stability_pLDA |
| pLR.m | Perform penalised linear regression. | [A]=pLR(X,y,lambda) | classif_pLR_loo stability_pLDA |
| stability_pLDA.m | Perform feature selection using penalised linear discriminant analysis with resampling. | [perc,nit] = stability_pLDA(X,y, B,lambda,thresh) | classif_pLDA_loo |
| stability_pLR.m | Perform feature selection using penalised linear regression with resampling. | [perc] = stability_pLR (X,y,B,lambda) | classif_pLR_loo |

Tab. 2: A list of input and output data and variables of the functions in the Penalised Reduction & Classification Toolbox.

| **Variable** | **Description** | **Used as input variable by** | **Used as output variable by** |
|---|---|---|---|
| A | matrix containing 0 and 1 (1..feature was selected); number of rows of A is equal to length(lambda) and number of columns is equal to number of features | | pLR |
| alpha | threshold for p-values calculated by t-test (and potentially corrected using FDR correction) | classif_Ttest_loo<br>classif_TtestFDR_loo | |
| B | number of iterations in resampling | classif_pLDA_loo<br>classif_pLR_loo<br>stability_pLDA<br>stability_pLR | |
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) | cperformance | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo<br>classifiers |
| class2 | matrix of votes (group identifiers) which are results of 2 classification of each person (rows are classifiers, columns are persons) | mcnemar | |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo<br>classifiers | |
| effic | efficiency of classifiers (i.e. classification performance) | | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo<br>cperformance |
| fsel | how many times features were selected as most discriminative ones | | classif_pLDA_loo<br>classif_pLR_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo |

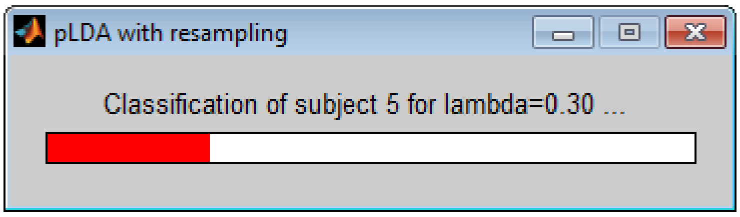
| **Variable** | **Description** | **Used as input variable by** | **Used as output variable by** |
|---|---|---|---|
| gis | vector with ground truth (true group identifiers) | cperformance<br>mcnemar | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo |
| lambda | regularization parameter | classif_pLDA_loo<br>classif_pLR_loo<br>pLDA<br>pLR<br>stability_pLDA<br>stability_pLR | |
| loo | binary variable:<br>0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV;<br>1..LOOCV is used only during classification - incorrect LOOCV | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo | |
| m | number of eigenvectors which you want to discard (a possitive number means discarding m eigenvectors with smallest eigenvalues, a negative number means discarding m eigenvectors with largest eigenvalues and zero means discarding no eigenvectors); absolute value of m must be less than a number of subjects minus 1 | classif_isPCA_loo<br>isPCA_eig<br>isPCA_reduc | |
| m1 | mean vector of group 1 | pLDA | |
| m2 | mean vector of group 2 | pLDA | |
| nit | matrix with number of iterations of convergence of pLDA | | classif_pLDA_loo<br>stability_pLDA |
| niters | number of iterations until convergence of pLDA | | pLDA |
| nsel | number of selected features (in each iteration) | classif_Ttest_nsel_loo | classif_pLDA_loo<br>classif_pLR_loo |
| p | p-value calculated using McNemar's test for comparison of two classifiers | | mcnemar |
| pat | identifier of patients | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo<br>cperformance | |

| Variable | Description | Used as input variable by | Used as output variable by |
|---|---|---|---|
| perc | percentages of how often feature was selected to play role in pLDA or pLR model | | stability_pLDA<br>stability_pLR |
| prob | probability threshold(s) for selection of features | classif_pLDA_loo<br>classif_pLR_loo | |
| pvar | explained variability by isPCA | | isPCA_reduc<br>isPCA_eig |
| s | within variance vector | pLDA | |
| thresh | threshold to be used as an assessment of convergence | classif_pLDA_loo<br>pLDA<br>stability_pLDA | |
| V | isPCA projection matrix which consists of eigenvectors | | isPCA_reduc<br>isPCA_eig |
| v | regression coefficients of pLDA | | pLDA |
| X | data matrix (rows correspond to persons, columns to features) | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo<br>pLR<br>isPCA_reduc<br>stability_pLDA<br>stability_pLR | |
| Xc | centered data matrix (where rows correspond to persons and colums to features) | isPCA_eig | |
| Xmean | mean person image (mean over all features) | | isPCA_reduc |
| Xred | reduced data matrix by isPCA | | isPCA_reduc |
| Xtest | testing data matrix or vector (row(s) correspond to persons, columns to features) | classifiers | |
| Xtrain | training data matrix (rows correspond to persons, columns to features) | classifiers | |
| y | row vector with group identifiers | classif_pLDA_loo<br>classif_pLR_loo<br>classif_isPCA_loo<br>classif_Ttest_loo<br>classif_Ttest_nsel_loo<br>classif_TtestFDR_loo<br>pLR<br>stability_pLDA<br>stability_pLR | |
| ytrain | row vector with group identifiers of training data | classifiers | |

```
9 classif_pLDA_loo
      9 stability_pLDA
             9 pLDA
      9 classifiers
      9 cperformance

9 classif_pLR_loo
      9 stability_pLR
             9 pLR
      9 classifiers
      9 cperformance

9 classif_isPCA_loo
      9 isPCA_reduc
             9 isPCA_eig
      9 classifiers
      9 cperformance

9 classif_Ttest_loo
      9 classifiers
      9 cperformance

9 classif_Ttest_nsel_loo
      9 classifiers
      9 cperformance

9 classif_TtestFDR_loo
      9 classifiers
      9 cperformance

9 mcnemar
```

Fig. 1: Hierarchy of the functions in the Penalised Reduction & Classification Toolbox.

The functions involved in the Penalised Reduction & Classification Toolbox are described in full details in following chapters. The functions are in alphabetical order. The *classif_pLDA_LOO*, *classif_pLR_LOO*, *classif_isPCA_LOO*, *classif_Ttest_LOO*, *classif_Ttest_nsel_LOO*, and *classif_TtestFDR_LOO* functions display a waitbar (an example is in Fig. 2) for illustration of how many subjects/objects have been classified up to now.


pLDA with resampling

Classification of subject 5 for lambda=0.30 ...

Fig. 2: Waitbar displayed by classification functions. It shows a number of successfully classified subjects or objects.

# 1. CLASSIF_ISPCA_LOO

## Purpose

Perform classification of two groups of subjects/objects based on data reduced using intersubject principal component analysis (isPCA).

## Syntax

[effic,class,gis] = classif_isPCA_loo(X,y,pat,classif,loo,m)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| pat | identifier of patients |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |
| loo | binary variable: 0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV; 1..LOOCV is used only during classification - incorrect LOOCV |
| m | number of eigenvectors which you want to discard (a possitive number means discarding *m* eigenvectors with smallest eigenvalues, a negative number means discarding *m* eigenvectors with largest eigenvalues and zero means discarding no eigenvectors); absolute value of *m* must be less than a number of images minus one |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |
| class | matrix of votes (group identifiers) which are results of classification algorithms for each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |

## Example

[effic,class,gis]=classif_isPCA_loo(X,y,1,[1,2],0,10)

- identifier of patients (or a group with higher risk) is equal to 1
- both diagonal LDA and linear SVM are used for classification
- LOOCV is used during data reduction and classification
- 10 eigenvectors with smallest eigenvalues are discarded

## Description

The *classif_isPCA_loo* function consists of the data reduction based on calculating eigenvectors using isPCA, data classification and validation using LOOCV. The *classif_isPCA_loo* function calls the *isPCA_reduc* function for data reduction, the *classifiers* function for classification of the reduced data and the *cperformance* function for calculating the classification performance. There is an opportunity to choose if LOOCV is used during data reduction and classification or only during classification. LOOCV means that every subject/object (a row vector of the matrix **X**) is chosen as a testing image stepwise and the remaining *n*-1 subjects/objects are used for training the classifier. The *classif_isPCA_loo* function shows a waitbar (see an example waitbar in Fig. 2).

## Algorithm

If *loo=0* (it means LOOCV is used during data reduction and classification):

- Repeat for each subject/object:
  - Reduce the *n*-1 training subjects/objects using the *isPCA_reduc* function; the parameter *m* enables to choose how many eigenvectors are discarded during creation of a projection matrix of isPCA.
  - Use the results of the *isPCA_reduc* function for subtraction of a mean of the training data from the testing subject/object and for the subsequent reduction of the testing data.
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

If *loo=1* (it means LOOCV is used only during classification):

- Reduce data of all subjects/objects using the *isPCA_reduc* function; the parameter *m* enables to choose how many eigenvectors are discarded during creation of a projection matrix of isPCA.
- Repeat for each subject/object:
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

## Notes

The mathematical background of isPCA algorithm is described in Appendix C.

## References

To learn more about isPCA, see (Janousova et al., 2015; Demirci et al., 2008; Fukunaga, 1990).

## 2. CLASSIF_PLDA_LOO

### Purpose

Perform classification of two groups of subjects/objects based on data reduced using penalised linear discriminant analysis (pLDA) with resampling.

### Syntax

[effic,class,gis,nit,fsel,nsel] =
classif_pLDA_loo(X,y,pat,classif,loo,thresh,lambda,prob,B)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| pat | identifier of patients |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |
| loo | binary variable: 0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV; 1..LOOCV is used only during classification - incorrect LOOCV |
| thresh | threshold to be used as an assessment of convergence |
| lambda | regularization parameter |
| prob | probability threshold(s) for selection of features |
| B | number of iterations in resampling |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |
| nit | matrix with number of iterations of convergence of pLDA |
| fsel | how many times features were selected (in case of incorrect LOOCV, fsel is equal to percentages calculated using pLDA) |
| nsel | number of selected features (in each iteration) |

### Example

[effic,class,gis,nit,fsel,nsel]=classif_pLDA_loo(X,y,1,[1,2],0,1e-07,0.3,0.99,100)

- identifier of patients (or a group with higher risk) is equal to 1
- both diagonal LDA and linear SVM are used for classification
- LOOCV is used during data reduction and classification
- values of parameters for pLDA are: thresh=1e-07, lambda=0.3, prob=0.99
- number of iterations in resampling is 100

### Description

The *classif_pLDA_loo* function consists of the data reduction based on selection of most discriminative features using pLDA with resampling, data classification and validation using LOOCV. The *classif_pLDA_loo* function calls the *stability_pLDA* function for data reduction, the *classifiers* function for classification of the reduced data and the *cperformance* function for calculating the classification performance. There is an opportunity to choose if LOOCV is used during data reduction and classification or only during classification. LOOCV means that every subject/object (a row vector of the matrix **X**) is chosen as a testing image stepwise and the remaining *n*-1 subjects/objects are used for training the classifier. The *classif_pLDA_loo* function shows a waitbar (see an example waitbar in Fig. 2).

## Algorithm

If *loo=0* (it means LOOCV is used during data reduction and classification):

- Repeat for each subject/object:
  - Calculate selection probabilities of features using the *stability_pLDA* function based on the *n*-1 training subjects/objects and defined parameters *B*, *lambda* and *thresh*.
  - For each value of *lambda* and *prob* select features with selection probability ≥ *prob* (i.e. perform data reduction) and then call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

If *loo=1* (it means LOOCV is used only during classification):

- Calculate selection probabilities of features using the *stability_pLDA* function based on all subjects/objects and defined parameters *B*, *lambda* and *thresh*.
- Repeat for each value of *lambda* and *prob* and for each subject/object:
  - Select features with selection probability ≥ *prob* (i.e. perform data reduction).
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

Calculate the classification performance by comparing the estimated group identifiers stored in matrix *class* with true group labels.

## Notes

The mathematical background of pLDA with resampling is described in Appendix A.

## References

For more information about pLDA with resampling, see (Janousova et al., under review).

## 3. CLASSIF_PLR_LOO

### Purpose

Perform classification of two groups of subjects/objects based on data reduced using penalised linear resampling (pLR) with resampling.

### Syntax

[effic,class,gis,fsel,nsel] = classif_pLR_loo(X,y,pat,classif,loo,lambda,prob,B)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| pat | identifier of patients |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |
| loo | binary variable: 0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV; 1..LOOCV is used only during classification - incorrect LOOCV |
| lambda | regularization parameter |
| prob | probability threshold(s) for selection of features |
| B | number of iterations in resampling |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |
| fsel | how many times features were selected (in case of incorrect LOOCV, fsel is equal to percentages calculated using pLR) |
| nsel | number of selected features (in each iteration) |

### Example

[effic,class,gis,fsel,nsel]=classif_pLR_loo(X,y,1,[1,2],0,0.3,0.5,100)

- identifier of patients (or a group with higher risk) is equal to 1
- both diagonal LDA and linear SVM are used for classification
- LOOCV is used during data reduction and classification
- values of parameters for pLR method are: lambda=0.3, prob=0.5
- number of iterations in resampling is 100

### Description

The *classif_pLR_loo* function consists of the data reduction based on selection of most discriminative features using pLR with resampling, data classification and validation using LOOCV. The *classif_pLR_loo* function calls the *stability_pLR* function for data reduction, *classifiers* function for classification of the reduced data and the *cperformance* function for calculating the classification performance. There is an opportunity to choose if LOOCV is used during data reduction and classification or only during classification. LOOCV means that every subject/object (a row vector of the matrix **X**) is chosen as a testing image stepwise and the remaining *n*-1 subjects/objects are used for training the classifier. The *classif_pLR_loo* function shows a waitbar (see an example waitbar in Fig. 2).

## Algorithm

If *loo=0* (it means LOOCV is used during data reduction and classification):

- Repeat for each subject/object:
    - Calculate selection probabilities of features using the *stability_pLR* function based on the *n*-1 training subjects/objects and defined parameters *B* and *lambda*.
    - For each value of *lambda* and *prob* select features with selection probability ≥ *prob* (i.e. perform data reduction) and then call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

If *loo=1* (it means LOOCV is used only during classification):

- Calculate selection probabilities of features using the *stability_pLR* function based on all subjects/objects and defined parameters *B* and *lambda*.
- Repeat for each value of *lambda* and *prob* and for each subject/object:
    - Select features with selection probability ≥ *prob* (i.e. perform data reduction).
    - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

Calculate the classification performance by comparing the estimated group identifiers stored in matrix *class* with true group labels.

## Notes

The mathematical background of pLR with resampling is described in Appendix B.

## References

For more information about pLR with resampling, see (Janousova et al. 2012).

# 4. CLASSIF_TTEST_LOO

## Purpose

Perform classification of two groups of subjects/objects based on data reduced using t-test with threshold alpha.

## Syntax

[effic,class,gis,fsel] = classif_Ttest_loo(X,y,pat,classif,alpha,loo)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| pat | identifier of patients |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |
| alpha | threshold for p-values calculated by t-test |
| loo | binary variable: 0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV; 1..LOOCV is used only during classification - incorrect LOOCV |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |
| fsel | how many times feature was selected as statistically significant |

## Example

[effic,class,gis,fsel]=classif_Ttest_loo(X,y,1,[1,2],0.05,0)

- identifier of patients (or a group with higher risk) is equal to 1
- both diagonal LDA and linear SVM are used for classification
- alpha=0.05
- LOOCV is used during data reduction and classification

## Description

The *classif_Ttest_loo* function consists of the data reduction based on selection of most discriminative features using two-sample t-test (i.e. features with p-value smaller than alpha), data classification and validation using LOOCV. The *classif_Ttest_loo* function calls the *classifiers* function for classification of the reduced data and the *cperformance* function for calculating the classification performance. There is an opportunity to choose if LOOCV is used during data reduction and classification or only during classification. LOOCV means that every subject/object (a row vector of the matrix **X**) is chosen as a testing image stepwise and the remaining *n*-1 subjects/objects are used for training the classifier. The *classif_Ttest_loo* function shows a waitbar (see an example waitbar in Fig. 2).

## Algorithm

If *loo=0* (it means LOOCV is used during data reduction and classification):

- Repeat for each subject/object:
    - Set the subject/object as the testing one; all remaining subjects/objects are the training data.
    - Perform the two-sample t-test using the *n*-1 training subjects/objects.
    - Reduce data by selecting features with p-value smaller than alpha.
    - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

If *loo=1* (it means LOOCV is used only during classification):

- Perform the two-sample t-test using all data.
- Reduce data by selecting features with p-value smaller than alpha.
- Repeat for each subject/object:
    - Set the subject/object as the testing one; all remaining subjects/objects are the training data.
    - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

Calculate the classification performance by comparing the estimated group identifiers stored in matrix *class* with true group labels.

## Notes

The function calls the function *ttest2* from the Statistics and Machine Learning Toolbox.

## 5. CLASSIF_TTEST_NSEL_LOO

**Purpose**

Perform classification of two groups of subjects/objects based on data reduced using selection of features with lowest p-values based on t-test.

**Syntax**

[effic,class,gis,fsel] = classif_Ttest_nsel_loo(X,y,pat,classif,nsel,loo)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| pat | identifier of patients |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |
| nsel | number of most significant features ordered based on t-test |
| loo | binary variable: 0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV; 1..LOOCV is used only during classification - incorrect LOOCV |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |
| fsel | how many times feature was selected as statistically significant |

**Example**

[effic,class,gis,fsel]=classif_Ttest_nsel_loo(X,y,1,[1,2],400,0)

- identifier of patients (or a group with higher risk) is equal to 1
- both diagonal LDA and linear SVM are used for classification
- 400 features with lowest p-value based on t-test are selected
- LOOCV is used during data reduction and classification

**Description**

The *classif_Ttest_nsel_loo* function consists of the data reduction based on selection of *nsel* most discriminative features, i.e. features with lowest p-values calculated using two-sample t-test, data classification and validation using LOOCV. The *classif_Ttest_nsel_loo* function calls the *classifiers* function for classification of the reduced data and the *cperformance* function for calculating the classification performance. There is an opportunity to choose if LOOCV is used during data reduction and classification or only during classification. LOOCV means that every subject/object (a row vector of the matrix **X**) is chosen as a testing image stepwise and the remaining *n*-1 subjects/objects are used for training the classifier. The *classif_Ttest_nsel_loo* function shows a waitbar (see an example waitbar in Fig. 2).

**Algorithm**

If *loo=0* (it means LOOCV is used during data reduction and classification):

- Repeat for each subject/object:
  - Set the subject/object as the testing one; all remaining subjects/objects are the training data.
  - Perform the two-sample t-test using the *n*-1 training subjects/objects.
  - Reduce data by selecting *nsel* features with lowest p-values.
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

If *loo=1* (it means LOOCV is used only during classification):

- Perform the two-sample t-test using all data.
- Reduce data by selecting *nsel* features with lowest p-values.
- Repeat for each subject/object:
  - Set the subject/object as the testing one; all remaining subjects/objects are the training data.
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

Calculate the classification performance by comparing the estimated group identifiers stored in matrix *class* with true group labels.

**Notes**

The function calls the function *ttest2* from the Statistics and Machine Learning Toolbox.

# 6. CLASSIF_TTESTFDR_LOO

## Purpose

Perform classification of two groups of subjects/objects based on data reduced using t-test with false discovery rate (FDR) correction.

## Syntax

[effic,class,gis,fsel] = classif_TtestFDR_loo(X,y,pat,classif,alpha,loo)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| pat | identifier of patients |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |
| alpha | threshold for p-values calculated by t-test and corrected using FDR correction |
| loo | binary variable: 0..leave-one-out cross-validation (LOOCV) is used during data reduction and classification - correct LOOCV; 1..LOOCV is used only during classification - incorrect LOOCV |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |
| fsel | how many times feature was selected as statistically significant |

## Example

[effic,class,gis,fsel]=classif_TtestFDR_loo(X,y,1,[1,2],0.05,0)

- identifier of patients (or a group with higher risk) is equal to 1
- both diagonal LDA and linear SVM are used for classification
- alpha=0.05
- LOOCV is used during data reduction and classification

## Description

The *classif_TtestFDR_loo* function consists of the data reduction based on selection of most discriminative features using the two-sample t-test (i.e. features with p-values corrected using FDR correction which are smaller than alpha), data classification and validation using LOOCV. The *classif_TtestFDR_loo* function calls the *classifiers* function for classification of the reduced data and the *cperformance* function for calculating the classification performance. There is an opportunity to choose if LOOCV is used during data reduction and classification or only during classification. LOOCV means that every subject/object (a row vector of the matrix **X**) is chosen as a testing image stepwise and the remaining *n*-1 subjects/objects are used for training the classifier. The *classif_TtestFDR_loo* function shows a waitbar (see an example waitbar in Fig. 2).

## Algorithm

If *loo=0* (it means LOOCV is used during data reduction and classification):

- Repeat for each subject/object:
  - Set the subject/object as the testing one; all remaining subjects/objects are the training data.
  - Perform the two-sample t-test using the *n*-1 training subjects/objects.
  - Correct calculated p-values using FDR correction.
  - Reduce data by selecting features with corrected p-value smaller than alpha.
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

If *loo=1* (it means LOOCV is used only during classification):

- Perform the two-sample t-test using all data.
- Correct calculated p-values using FDR correction.
- Reduce data by selecting features with corrected p-value smaller than alpha.
- Repeat for each subject/object:
  - Set the subject/object as the testing one; all remaining subjects/objects are the training data.
  - Call the *classifiers* function to train chosen classifier(s) using reduced training data, classify the reduced testing subject/object based on the trained classifier(s) and store the estimated group identifier into the matrix *class*.

Calculate the classification performance by comparing the estimated group identifiers stored in matrix *class* with true group labels.

## Notes

The function calls the function *ttest2* from the Statistics and Machine Learning Toolbox and the function *FDR.m* (http://www-personal.umich.edu/~nichols/FDR/).

# 7. CLASSIFIERS

**Purpose**

Classify reduced data using diagonal LDA or linear SVM classifiers.

**Syntax**

[class] = classifiers(Xtest,Xtrain,ytrain,classif)

Input data and variables:

| | |
|---|---|
| Xtest | testing data matrix or vector (row(s) correspond to persons, columns to features) |
| Xtrain | training data matrix (rows correspond to persons, columns to features) |
| ytrain | row vector with group identifiers of training data |
| classif | vector of classifiers which are used (1..diagonal LDA, 2..linear SVM); for example: [1,2] - both classifiers are used for classification; [2] - linear SVM is used for classification) |

Output data and variables:

| | |
|---|---|
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |

**Description**

The *classifiers* function is called by *classif_pLDA_LOO*, *classif_pLR_LOO*, *classif_isPCA_LOO*, *classif_Ttest_LOO*, *classif_Ttest_nsel_LOO*, and *classif_TtestFDR_LOO* functions. Currently, it enables classification of reduced data using two classifiers, namely diagonal LDA or linear SVM. Newertheless, it can be easily extended to allow for classification by other classification methods.

**Algorithm**

- Choose which one of the classifiers will be used for classification based on the parameter *classif*.
- Train the classifier using training data and perform classification of testing data.

**Notes**

The function calls the functions *classify*, *svmtrain* and *svmclassify* from the Statistics and Machine Learning Toolbox. The function can be easily extended to incorporate more classifiers.

# 8. CPERFORMANCE

## Purpose

Estimate classification performance.

## Syntax

[effic] = cperformance(class,gis,pat)

Input data and variables:

| | |
|---|---|
| class | matrix of votes (group identifiers) which are results of classification of each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |
| pat | identifier of patients |

Output data and variables:

| | |
|---|---|
| effic | efficiency of classifiers (i.e. classification performance) |

## Description

The *cperformance* function computes classification performance measures to evaluate the efficiency of classifiers. An output variable *effic* is a matrix where rows correspond to classifiers and columns are true positive (TP) results, false negative (FN) results, true negative (TN) results, false positive (FP) results, accuracy, sensitivity, specificity and precision.

## Algorithm

- Compute classification performance of each classifier, specifically:
  - accuracy = (TP+TN) / (TP+TN+FP+FN)
  - sensitivity = TP / (TP+FN)
  - specificity = TN / (TN+FP)
  - precision = TP / (TP+FP)

## References

To learn more about the evaluation of efficiency of classifiers, see (Altman, 1999).

# 9. ISPCA_EIG

**Purpose**

Compute projection matrix of isPCA.

**Syntax**

[V,pvar] = isPCA_eig(Xc,m)

Input data and variables:

| | |
|---|---|
| Xc | centered data matrix (where rows correspond to persons and colums to features) |
| m | number of eigenvectors which you want to discard (a possitive number means discarding *m* eigenvectors with smallest eigenvalues, a negative number means discarding *m* eigenvectors with largest eigenvalues and zero means discarding no eigenvectors); absolute value of *m* must be less than a number of images minus one |

Output data and variables:

| | |
|---|---|
| V | isPCA projection matrix which consists of eigenvectors |
| pvar | explained variability by isPCA |

**Description**

The *isPCA_eig* function is a subordinate function of the *isPCA_reduc* function. It is the core function of intersubject PCA. A user can choose a number of eigenvectors *m* which are discarded during creation of a projection matrix **V** of isPCA. The parameter *m* influences the explained variability *pvar*.

**Algorithm**

- Compute the covariance matrix of subjects and its eigenvectors and eigenvalues.
- Sort eigenvalues from largest to smallest, choose non-zero eigenvalues and sort and choose the corresponding eigenvectors.
- Discard *m* eigenvectors.
- Transform eigenvectors of the covariance matrix of subjects into eigenvectors of the covariance matrix of features and save them as columns into the projection matrix **V**.

**Notes**

The mathematical background of isPCA is described in Appendix C.

**References**

To learn more about the computation of eigenvectors of the covariance matrix of features from eigenvectors of the covariance matrix of persons, see (Janousova et al., 2015; Demirci et al., 2008; Fukunaga, 1990).

# 10. ISPCA_REDUC

**Purpose**

Reduce data using isPCA.

**Syntax**

[Xred,pvar,Xmean,V] = isPCA_reduc(X,m)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| m | number of eigenvectors which you want to discard (a possitive number means discarding *m* eigenvectors with smallest eigenvalues, a negative number means discarding *m* eigenvectors with largest eigenvalues and zero means discarding no eigenvectors); absolute value of *m* must be less than a number of images minus one |

Output data and variables:

| | |
|---|---|
| Xred | reduced data matrix by isPCA |
| pvar | explained variability by isPCA |
| Xmean | mean person image (mean over all features) |
| V | isPCA projection matrix which consists of eigenvectors |

**Description**

The *isPCA_reduc* function is a subordinate function of the *classif_isPCA_loo* function. It serves for reduction of data matrix **X** using the projection matrix **V** of isPCA which is calculated by calling the *isPCA_eig* function. A number of eigenvectors *m* to be discarded is passed to the *isPCA_eig* function. The results of the *isPCA_reduc* function are the reduced data matrix *Xred*, explained varialibity *pvar* and two variables important for reduction of a testing image. They are a mean person image *Xmean* which is subtracted from the testing image and the isPCA projection matrix **V** which is used for reduction of the testing image.

**Algorithm**

- Compute a mean person image *Xmean* and subtract it from all rows of the data matrix **X**.
- Call the function *isPCA_eig* and reduce the centered matrix $\mathbf{X}_c$ using the projection matrix **V**.

**Notes**

The mathematical background of isPCA is described in Appendix C.

**References**

To learn more about isPCA, see (Janousova et al., 2015; Demirci et al., 2008; Fukunaga, 1990).

# 11. MCNEMAR

**Purpose**

Compute p-value using McNemar’s test for comparison of two classifiers.

**Syntax**

[p]=mcnemar(class2,gis)

Input data and variables:

| | |
|---|---|
| class2 | matrix of votes (group identifiers) which are results of 2 classification algorithms for each person (rows are classifiers, columns are persons) |
| gis | vector with ground truth (true group identifiers) |

Output data and variables:

| | |
|---|---|
| p | p-value calculated using McNemar's test for comparison of two classifiers |

**Description**

The *mcnemar* function compares classification performance of two classifiers. An input into the function is a matrix of votes (estimated group identifiers) which are calculated using functions *classif_pLDA_LOO, classif_pLR_LOO, classif_isPCA_LOO, classif_Ttest_LOO, classif_Ttest_nsel_LOO,* or *classif_TtestFDR_LOO.*

**Algorithm**

- Calculate a number of subjects incorrectly classified by the first classifier but correctly classified by the second classifier ($N_{01}$).
- Calculate a number of subjects correctly classified by the first classifier but misclassfied by the second one ($N_{10}$).
- Compute $\chi^2$ statistics using following formula: $\chi^2 = \frac{(|n_{01}-n_{10}|-1)^2}{n_{01}+n_{10}}$.
- Estimate a p-value based on the $\chi^2$ statistics.

**References**

Computation of p-value using McNemar’s test is peformed based on the formula 1.11 from (Kuncheva 2004).

## 12. PLDA

### Purpose

Perform penalised linear discriminant analysis (pLDA).

### Syntax

[v,niters] = pLDA(m1, m2,s,lambda,thresh)

Input data and variables:

| | |
|---|---|
| m1 | mean vector of group 1 |
| m2 | mean vector of group 2 |
| s | within variance vector |
| lambda | regularization parameter |
| thresh | threshold to be used as an assessment of convergence |

Output data and variables:

| | |
|---|---|
| v | regression coefficients of pLDA |
| niters | number of iterations until convergence of pLDA |

### Description

The *pLDA* function is a subordinate function of the *stability_pLDA* function. It is the core function of the pLDA method. The aim is to estimate coefficients of a direction vector which discriminates between two groups of subjects such that some of the coefficients are set to zero. The amount of zero coefficients is controlled by the regularization parameter *lambda*. When *lambda* is zero, no penalty is imposed on the coefficients (i.e. no data reduction is performed). As *lambda* increases from zero, more coefficients are set to zero; and finally, at the maximum value of *lambda*, all coefficients are set to zero. The pLDA algorithm works in iterative manner and the parameter *thresh* regulates the convergence of the algorithm.

### Algorithm

```
· Identify and remove features with zero variance.
· Initialize $\mathbf{v}^0 = \mathbf{S}_W^{*-\mathbf{1}}(\mathbf{m}_H - \mathbf{m}_D)^T$.
· Normalize $\mathbf{v}^0$ such that $\mathbf{v}^{0T}\mathbf{S}_W^*\mathbf{v}^0 = 1$.
· Repeat
    ○ set $\mathbf{a} = \mathbf{S}_W^{*-\mathbf{1}}\left((\mathbf{m}_H - \mathbf{m}_D)(\mathbf{m}_H - \mathbf{m}_D)^T\mathbf{v}^0\right)$
    ○ calculate $S_\lambda(\mathbf{a}) = \mathrm{sign}(\mathbf{a})(|\mathbf{a}| - \lambda)_+$, where $(\cdot)_+ = \max(0,\cdot)$
    ○ compute $\hat{\mathbf{v}} = \mathbf{S}_W^{*-\mathbf{1}}(S_\lambda(\mathbf{a})/\|S_\lambda(\mathbf{a})\|_\mathbf{2})$
    ○ set $\mathbf{v}^0 = \hat{\mathbf{v}}$
· until $\hat{\mathbf{v}}$ converges; specifically, until $\left|1 - \left|\sum_{j=1}^{p}\hat{v}_j v_j^0 s_j^2\right|\right| < 10^{-7}$.
```

### Notes

The mathematical background of pLDA is described in Appendix A. More details about the pLDA method can be found in (Janousova et al., under review).

# 13. PLR

## Purpose

Perform penalised linear regression (pLR).

## Syntax

[A] = pLR(X,y,lambda)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| lambda | regularization parameter |

Output data and variables:

| | |
|---|---|
| A | matrix containing 0 and 1 (1..feature was selected); number of rows of A is equal to length(lambda) and number of columns is equal to number of features |

## Description

The *pLR* function is a subordinate function of the *stability_pLR* function. It is the core function of the pLR method. The aim is to estimate regression coefficients such that the model will enable selection of features most associated with the vector **y** while correlated features are selected in groups. The amount of selected features is controlled by the regularization parameter *lambda*. When *lambda* is zero, no penalty is imposed on the coefficients (i.e. no data reduction is performed). As *lambda* increases from zero, more coefficients are set to zero and thus less features are selected; and finally, at the maximum value of *lambda*, all coefficients are set to zero and no features are selected.

## Algorithm

- Standardize columns of the matrix **X**.
- Perform mean-centering of the vector **y**.
- Identify and remove features with zero variance.
- Calculate the soft thresholded coefficient vector $\hat{\boldsymbol{\beta}} = \mathrm{sign}(\mathbf{X}^T\mathbf{y})\,(|\mathbf{X}^T\mathbf{y}| - \lambda/2)_+$, where $(\cdot)_+ = \max(0,\cdot)$.
- Identify features with non-zero coefficients and store the information in a matrix **A**.

## Notes

The mathematical background of pLR is described in Appendix B. More details about the pLR method can be found in (Janousova et al., 2012).

## 14. STABILITY_PLDA

**Purpose**

Perform feature selection using penalised linear discriminant analysis (pLDA) with resampling.

**Syntax**

[perc,nit] = stability_pLDA(X,y,B,lambda,thresh)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| B | number of iterations in resampling |
| lambda | regularization parameter |
| thresh | threshold to be used as an assessment of convergence |

Output data and variables:

| | |
|---|---|
| perc | percentages of how often feature was selected to play role in pLDA model |
| nit | matrix with number of iterations of convergence of pLDA |

**Description**

The *stability_pLDA* function is a subordinate function of the *classif_pLDA_loo* function. It serves for selection of most discriminative features using pLDA (performed by calling the *pLDA* function) with resampling. The result of the *stability_pLDA* function is the matrix *perc* (rows correspond to values of lambda, columns to features) containing selection probabilities, i.e. percentages of how often a feature is selected as a discriminative one (i.e. its corresponding coefficient of direction vector calculated using pLDA is non-zero) in *B* random subsamples of data.

**Algorithm**

- Repeat *B*-times:
  - Select a subset comprising a half of subjects from both groups randomly.
  - Calculate mean vectors for each group and a within variance vector.
  - Compute coefficients of a direction vector which discriminates between two groups of subjects using pLDA algorithm.
  - Create indicator variable showing which features correspond to non-zero coefficients.
- Calculate the percentages *perc* by dividing the sum of indicator variables by the number of iterations *B*.

**Notes**

The mathematical background of pLDA is described in Appendix A. More details about the pLDA method can be found in (Janousova et al., under review).

## 15. STABILITY_PLR

**Purpose**

Perform feature selection using penalised linear regression (pLR) with resampling.

**Syntax**

[perc] = stability_pLR(X,y,B,lambda)

Input data and variables:

| | |
|---|---|
| X | data matrix (rows correspond to persons, columns to features) |
| y | row vector with group identifiers |
| B | number of iterations in resampling |
| lambda | regularization parameter |

Output data and variables:

| | |
|---|---|
| perc | percentages of how often feature was selected to play role in pLR model |

**Description**

The *stability_pLR* function is a subordinate function of the *classif_pLR_loo* function. It serves for selection of most discriminative features using pLR (performed by calling the *pLR* function) with resampling. The result of the *stability_pLR* function is the matrix *perc* (rows correspond to values of lambda, columns to features) containing selection probabilities, i.e. percentages of how often a feature is selected as a discriminative one (i.e. its corresponding regression coefficient calculated using pLR is non-zero) in *B* random subsamples of data.

**Algorithm**

- Repeat *B*-times:
  - Select a subset comprising a half of subjects from both groups randomly.
  - Compute a matrix containing indicators showing which features correspond to non-zero coefficients calculated using pLR algorithm.
- Calculate the percentages *perc* by dividing the sum of indicators by the number of iterations *B*.

**Notes**

The mathematical background of pLR is described in Appendix B. More details about the pLR method can be found in (Janousova et al. 2012).

# APPENDIX

## APPENDIX A Mathematical background of penalised linear discriminant analysis (pLDA) with resampling

An input into pLDA with resampling is a $(n \times p)$ feature matrix **X** containing $p$-dimensional row vectors of features for each of $n$ independent subject and a $(n \times 1)$ vector **y** with labels of the subjects. Specifically, $y_i = 1$ if individual $i$ belongs to class D (diseased) and $y_i = 0$ when individual $i$ is from class H (healthy controls). The number of individuals in each class is $n_D$ and $n_H$, respectively, and $n = n_D + n_H$.

In pLDA, a lasso penalty is imposed on the $l_1$ norm of the direction vector **v** (Witten & Tibshirani 2011) that best discriminates two classes within a data sample via maximizing the between-class variance and simultaneous minimizing of the within-class variance. Imposing the lasso penalty leads to setting the coefficients $v_j$ , $j = 1, \dots, p$, of the least discriminative features to zero. The direction vector **v** is a solution of the following optimization problem:

$$\max_{\boldsymbol{v}} \{ \mathbf{v}^T \mathbf{S}_B \mathbf{v} - \lambda \textstyle\sum_{j=1}^{p} s_j |v_j| \} \text{ subject to } \mathbf{v}^T \mathbf{S}_W^* \mathbf{v} = 1,$$

where $\mathbf{S}_B = (\mathbf{m}_H - \mathbf{m}_D)^T (\mathbf{m}_H - \mathbf{m}_D)$ is the between-class scatter matrix; $\mathbf{S}_W^*$ is the diagonal estimate of the within-class scatter matrix $\mathbf{S}_W = \sum_{i=1}^{n_H} (\mathbf{x}_{i.} - \mathbf{m}_H)^T (\mathbf{x}_{i.} - \mathbf{m}_H) + \sum_{i=1}^{n_D} (\mathbf{x}_{i.} - \mathbf{m}_D)^T (\mathbf{x}_{i.} - \mathbf{m}_D)$; $\mathbf{m}_H = 1/n_H \cdot \sum_{i=1}^{n_H} \mathbf{x}_{i.}$ is the mean vector of class *H* (healthy controls); $\mathbf{m}_D = 1/n_D \cdot \sum_{i=1}^{n_D} \mathbf{x}_{i.}$ is the mean vector of class *D* (diseased individuals); $n_H$ and $n_D$ are numbers of subjects in class *H* and *D*, respectively; $\mathbf{x}_{i.}$, $i = 1, \dots, n$, are rows of the feature matrix **X**; and $\lambda$ is a regularization parameter that controls the amount of sparsity in the model. Specifically, when $\lambda$ is exactly zero, no penalty is imposed and all $p$ features contribute in the direction vector **v**. As $\lambda$ increases from zero, less features contribute in **v**. At its maximum value, all coefficients of **v** are set to zero.

The pLDA algorithm is combined with a resampling method proposed in (Meinshausen & Bühlmann 2010) for sparse predictive modelling. This procedure aims to calculate selection probabilities $P_j(\lambda)$ for each feature by repeatedly fitting the pLDA model on random subsets of the data set, while keeping track of the features associated to non-zero coefficients of **v**. Specifically, an indicator variable $\mathbf{c}^{(b)}(\lambda) = \left( c_1^{(b)}(\lambda), \dots, c_p^{(b)}(\lambda) \right)$ is calculated in each iteration $b = 1, \dots, B$, where $c_j^{(b)}(\lambda)$, $j = 1, \dots, p$, is equal to 1 if the $j$th feature is selected (i.e. its respective coefficient $v_j$ is non-zero) and 0 otherwise. Using all $B$ random subsets, the selection probability for each feature is then calculated as:

$$P_j(\lambda) = \tfrac{1}{B} \textstyle\sum_{b=1}^{B} c_j^{(b)}(\lambda), j = 1, \dots, p.$$

The final set of the most discriminative features consists of features with selection probability higher than a predefined threshold, e.g. 0.99.

## APPENDIX B Mathematical background of penalised linear regression (pLR) with resampling

An input into pLR with resampling is a $(n \times p)$ feature matrix $\mathbf{X}$ containing $p$-dimensional row vectors of features for each of $n$ independent subject and a $(n \times 1)$ vector $\mathbf{y}$ with labels of the subjects.

Prior to performing pLR, the columns of the matrix $\mathbf{X}$ are standardized (i.e. forced to have zero mean and unit variance) and the vector $\mathbf{y}$ is mean-centered. The standardized $\mathbf{X}$ and mean-centered $\mathbf{y}$ and are inputs into the regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, in which $\mathbf{y}$ is treated as a response variable, columns of $\mathbf{X}$ (i.e. feature vectors) are predictors and $\boldsymbol{\varepsilon}$ is an error variable. The aim is to estimate the regression coefficients $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$ such that the model will enable selection of imaging features most associated with the vector $\mathbf{y}$ (i.e. most discriminative features in case of binary $\mathbf{y}$ containing class labels) while correlated features are selected in groups by using the elastic net penalty (Zou & Hastie, 2005). The elastic net penalty contains two additive penalty terms: $l_1$ (lasso) penalty which enables selection of the features by inducing sparse solutions (Tibshirani 1996), and $l_2$ (ridge) penalty inducing a grouping effect on correlated variables (Zou & Hastie, 2005; Hoerl & Kennard, 1970). The elastic net estimates are found by minimizing the following least squares problem:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\kappa\|_2^2 + \lambda\|\boldsymbol{\beta}\kappa\|_1 + \mu\|\boldsymbol{\beta}\kappa\|_2^2\},$$

where $\lambda > 0$ and $\mu > 0$ are regularization parameters introduced for the $l_1$ and $l_2$ penalties, respectively. The parameter $\lambda$ controls the amount of sparsity (i.e. the number of selected features), specifically when the parameter $\lambda$ is zero, no penalty is imposed and all $p$ features are selected. For larger values of $\lambda$, sparser solutions are acquired, and for the maximum value of $\lambda$, no features are selected. The parameter $\mu$ regulates the amount of smoothing imposed on the regression coefficients associated with correlated features. The scaling factor $\kappa = (1 + \mu)^{-1}$ enables to correct for the double shrinkage caused by applying both penalties.

According to (Zou & Hastie, 2005), when $\mu$ is set to infinity, the number of tuned parameters is reduced down to only one, $\lambda$, while still maintaining the grouping effect. By setting $\mu$ to infinity, the term to be minimized is thus simplified to:

$$-2\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1.$$

It leads to a computationally cheap estimation algorithm, in which regression coefficients are estimated as:

$$\hat{\beta}_j = \operatorname{sign}(\mathbf{x}_{.j}^T\mathbf{y})\left(|\mathbf{x}_{.j}^T\mathbf{y}| - {}^{\lambda}\!/_{2}\right)_+,$$

where $(\cdot)_+$ is defined as $\max(0,\cdot)$ and $\mathbf{x}_{.j}^T,\ j = 1, \dots, p$ are column vectors in the matrix $\mathbf{X}$.

The pLR algorithm is combined with a resampling method proposed in (Meinshausen & Bühlmann 2010) for sparse predictive modelling. This procedure aims to calculate selection probabilities $P_j(\lambda)$ for each feature by repeatedly fitting the pLR model on random subsets of the data set, while keeping track of the features associated to non-zero coefficients of $\hat{\boldsymbol{\beta}}$.

Specifically, an indicator variable $\mathbf{c}^{(b)}(\lambda) = \left(c_1^{(b)}(\lambda), \dots, c_p^{(b)}(\lambda)\right)$ is calculated in each iteration $b = 1, \dots, B$, where $c_j^{(b)}(\lambda), j = 1, \dots, p$, is equal to 1 if the $j$th feature is selected (i.e. its respective coefficient $\hat{\beta}_j$ is non-zero) and 0 otherwise. Using all $B$ random subsets, the selection probability for each feature is then calculated as:

$$P_j(\lambda) = \frac{1}{B}\sum_{b=1}^{B} c_j^{(b)}(\lambda), j = 1, \dots, p.$$

The final set of the most discriminative features consists of features with selection probability higher than a predefined threshold, e.g. 0.5.

## APPENDIX C Mathematical background of intersubject principal component analysis (isPCA)

An input into the isPCA is $(n \times p)$ data matrix $\mathbf{X}$ which is composed of $n$ subjects described by $p$ features. According to linear algebra rules, nonzero eigenvalues of covariance matrix of features $\mathbf{X}^T\mathbf{X}$ and covariance matrix of subjects (intersubject covariance matrix) $\mathbf{X}\mathbf{X}^T$ are identical and eigenvectors corresponding to the higher dimensional covariance matrix can be derived from the eigenvectors of the smaller one by:

$$\mathbf{v}_j = \frac{\mathrm{X}^T\mathbf{w}_j}{\sqrt{\lambda_j(n-1)}}, j = 1, \dots, p.$$

where $\mathbf{v}_j$ is the $j$th eigenvector of the covariance matrix of features, $\mathbf{X}^T$ is the transposed image data matrix, $\mathbf{w}_j$ is the $j$th eigenvector of the intersubject covariance matrix, and $\lambda_j$ is the $j$th eigenvalue of the intersubject covariance matrix. The proof of the transformation is given in (Demirci et al., 2008; Fukunaga, 1990).

The isPCA algorithm can be described in following way:

1. Calculate $(n \times n)$ covariance matrix of subjects $\mathbf{C}_s$ of the data matrix $\mathbf{X}$ by $\mathbf{C}_s = (1/n - 1)(\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T$, where $n$ is the number of subjects and $\bar{\mathbf{X}}$ is a matrix with all rows equal to a mean image $\bar{\mathbf{x}}$ which is defined by $\bar{\mathbf{x}} = (1/n)\sum_{i=1}^{n}\mathbf{x}_i$, where $\mathbf{x}_i$, $i = 1, \dots, n$, are rows of the matrix $\mathbf{X}$.
2. Find $\lambda_j$ eigenvalues and $\mathbf{w}_j$ eigenvectors of the covariance matrix of subjects $\mathbf{C}_s$, $j = 1, \dots, n$.
3. Select $m$ eigenvectors (up to $n - 1$ eigenvectors that correspond to all nonzero eigenvalues).
4. Compute eigenvectors $\mathbf{v}_j$ of the covariance matrix of features $\mathbf{C}_\mathrm{v}$ by $\mathbf{v}_j = \frac{\mathrm{X}^T\mathbf{w}_j}{\sqrt{\lambda_j(n-1)}}$.
5. Construct $(p \times m)$ projection matrix $\mathbf{V}$ with column-wise computed eigenvectors $\mathbf{v}_j$.
6. Compute a reduced data matrix $\mathbf{Y}$ with the size of $(n \times m)$ by $\mathbf{Y} = (\mathbf{X} - \bar{\mathbf{X}}) \cdot \mathbf{V}$.

## REFERENCES

Altman D. G. 1999, *Practical Statistics for Medical Research*, Chapman and Hall/CRC, London.

Demirci O., Clark V. P., Magnotta V. A., Andreasen N. C., Lauriello J., Kiehl K. A., Pearlson G. D. & Calhoun V. D. 2008, A Review of challenges in the use of fMRI for disease classification / characterization and a projection pursuit application from multi-site fMRI schizophrenia study, *Brain Imaging and Behavior*; 2:147-226.

Fukunaga K. 1990, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego.

Hoerl A. & Kennard R. 1970, Ridge Regression - Applications to Nonorthogonal Problems. *Technometrics*; 12(1):69-82.

Janousova E., Montana G., Kasparek T. & Schwarz D., A supervised multivariate whole-brain reduction did not help to achieve high classification performance in schizophrenia research, Frontiers in Neuroscience, under review.

Janousova E., Schwarz D., Kasparek T. 2015, Combining various types of classifiers and features extracted from magnetic resonance imaging data in schizophrenia recognition. *Psychiatry Research: Neuroimaging*; 232(3): 237-249.

Janousova E., Vounou M., Wolz R., Gray K.R., Rueckert D., Montana G. & ADNI 2012, Biomarker discovery for sparse classification of brain images in Alzheimer's disease. *Annals of the BMVA*; 2:1-11.

Kuncheva L.I., 2004, *Combining Classifiers: Ideas and Methods*, John Wiley & Sons, Hoboken, NJ.

Meinshausen N. & Bühlmann P. 2010, Stability selection. *J R Stat Soc Ser B - Stat Methodol*; 72:417-473.

Tibshirani R. 1996, Regression shrinkage and selection via the Lasso. *J R Stat Soc Ser B-Methodol*; 58(1):267-288.

Witten D.M. & Tibshirani R 2011, Penalized classification using Fisher’s linear discriminant. *J R Stat Soc Ser B - Stat Methodol*; 73:753-772.

Zou H. & Hastie T. 2005, Regularization and variable selection via the elastic net. *J R Stat Soc Ser B-Stat Methodol*; 67:301-320.