Package ‘opls’

July 3, 2014

Title Implementation and extension of Orthogonal Projection to Latent Structures

Version 0.1

Description Implementation of orthogonal projection onto latent structures (OPLS). Based on the original method described by Trygg and Wold (Trygg J,Wold S. Orthogonal projections to latent structures (O-PLS). J.
Chemometrics 2002; 16: 119128). Includes novel variable selection and
n-group discriminant analysis. This software is provided ``AS IS'' without
warranty of any kind, express or implied.

Depends R (>= 3.1.0)

Imports pracma,caret

License MIT License

LazyData true

R topics documented:

apply_opls_model *apply_opls_model*

Description

Apply a model to new (or old) data. Computes the t scores and the predicted-y value for each sampel in new_X.

Usage

```
apply_opls_model(X, Y, opls_results, new_X)
```

Arguments

X	- n x p matrix, where n is the number of samples and p is the number of variables.
Y	- n x 1 matrix. Must be numeric
new_X	- m x p matrix, where m is the number of samples and p is the number of variables.
opls_model	- opls model.

Value

List containing

t	t-score
t_ortho	t-orthogonal scores
Y_pred	the predicted-y values for the samples

Examples

```
X <- rand(10,10)
new_X <- rand(5,10)
Y <- rand(10,1)
model <- opls(X,Y,1)
res <- apply_opls_model(X,Y,model,new_X)
```

determine_significant_features *determine_significant_features*

Description

Permute the values of each variable and recompute its loadings. Compare this distribution of loadings with the original loading for each variable.

Usage

```
determine_significant_features(X, Y, orig_model, num_permutations, talpha,
  inside_num_permutations, inside_talpha)
```

Arguments

`X`	- n x p matrix, where n is the number of samples and p is the number of variables.
`Y`	- n x 1 matrix. Must be numeric
`orig_model`	- original model.
`num_permutations`	- Number of permutation iterations.
`talpha`	- test alpha to use as cutoff
`inside_num_permutations`	- the smaller number of permutations used to efficiently rule out insignificant features. Should be less than num_permutations.
`inside_talpha`	- corresponding test alpha for ruling out loadings that are not close to being significant. This is purely for efficient calculations.

Value

List containing

`sig_inxs`	Significant indices
`not_sig_inxs`	Indices that are not significant
`significant`	0/1 bit vector for significant variables
`p_permuted`	the permuted loading values

Examples

```
X <- rand(10,10)
Y <- rand(10,1)
model <- opls(X,Y,1)
res <- determine_significant_features(X,Y,model,500,0.05,100,0.2)
```

`n.group.opls` *n.group.opls*

Description

Create an OPLS-DA model when there are more than 2 classes. This function is not applicable if you have a continuous response variable. It iteratively adds classes to the model. New classes are added such that they maximize the Q2. The class label is determine by the previous model (i.e., new data is projected into the previous model).

Usage

```
n.group.opls(X, Y, num_permutations, CV, nIterations = 100,
  min_num_OPLS_fact = 0)
```

Arguments

`X`	- n x p matrix, where n is the number of samples and p is the number of variables.
`Y`	- n x 1 matrix. Must be numeric
`num_permutations`	- number of permutation for the randomization test.
`CV`	- Parameter for internal cross-validation. -1 for leave-one-out cross-validation. The value of k in k-fold cross-validation otherwise.
`nIterations`	- number of iterations for external validation. One of each sample is held out each iteration.
`min_num_OPLS_fact`	- minimum number of OPLS factors. Default 0.

Value

List containing

`Q2`	External Q2 value
`helper.results`	Results from running helper function, including an opls model, opls model history, original unique Y values, new unique Y values, and adjusted Y values

Examples

```
X <- rand(10,10)
new_X <- rand(5,10)
Y <- rand(12,1)
Y[1:4,] = 1
Y[5:8,] = 2
Y[9:12,] = 3
res <- n.group.opls(X,Y,100,-1)
```

n.group.opls.helper *n.group.opls.helper*

Description

An internal helper function. Do not call this directly.

Usage

```
n.group.opls.helper(X, Y, num_permutations, CV, min_num_OPLS_fact = 0)
```

Arguments

`X`	- n x p matrix, where n is the number of samples and p is the number of variables.
`Y`	- n x 1 matrix. Must be numeric
`num_permutations`	- number of permutation for the randomization test.
`CV`	- number of folds for k-fold cross-validation or -1 for leave one out.
`min_num_OPLS_fact`	- minimum number of OPLS factors to consider.

opls *opls.*

Description

opls.

This allows you to create an OPLS model if you know the number of orthogonal components.

Usage

```
opls(X, Y, num_OPLS_fact)
```

Arguments

`X`	- n x p matrix, where n is the number of samples and p is the number of variables.
`Y`	- n x 1 matrix. Must be numeric
`num_OPLS_fact`	- Integer specifying the number of OPLS orthogonal components.

Value

the found opls model

Examples

```
model <- opls(rand(10,10),rand(10,1),1)
```

`opls_CV` *opls_CV*

Description

This allows you to create an OPLS model if you know the number of orthogonal components.

Usage

```
opls_CV(X, Y, num_OPLS_fact, folds)
```

Arguments

`X` - n x p matrix, where n is the number of samples and p is the number of variables.

`Y` - n x 1 matrix. Must be numeric

`num_OPLS_fact` - Integer specifying the number of OPLS orthogonal components.

`folds` - Number of k-fold cross-validation groups or -1 for leave one out cross-validation.

Value

List containing

`Q^2` Cross-validated R^2

`Q2s` One for each iteration

`press` residual calculation used in Q^2 calculation

`accuracy` standard accuracy that is only relevant if this is a classification problem

Examples

```
res <- opls_CV(rand(10,10),rand(10,1),1,-1)
```

`permutation_test` *permutation_test*

Description

Internal helper function. Do not use.

Usage

```
permutation_test(X, Y, num_OPLS_fact, num_permutations, folds)
```

Arguments

`X` - n x p matrix, where n is the number of samples and p is the number of variables.

`Y` - n x 1 matrix. Must be numeric

`num_OPLS_fact` - number of orthogonal OPLS factors

`num_permutations` - number of permutation for the randomization test.

`folds` - number of folds for CV

`run_det_sig` *run_det_sig*

Description

This is an internal helper function. Do not call directly.

Usage

```
run_det_sig(X, Y, orig_model, N, variables)
```

Arguments

`X` - n x p matrix, where n is the number of samples and p is the number of variables.

`Y` - n x 1 matrix. Must be numeric

`orig_model` - original model.

`N` - Number of permutation iterations.

`variables` - list of variables to test

`run_opls` *run_opls*

Description

This will perform OPLS analysis including cross-validation and a permutation test.

Usage

```
run_opls(X, Y, num_permutations, CV, min_num_OPLS_fact = 0)
```

Arguments

`X` - n x p matrix, where n is the number of samples and p is the number of variables.

`Y` - n x 1 matrix. Must be numeric

`num_permutations` - Number of permutation iterations.

`CV` - Number of k-fold cross-validation groups or -1 for leave one out cross-validation.

`min_num_OPLS_fact` - minimum number of OPLS orthogonal components. Default 0.

Value

List containing

`model` Final OPLS model

`Q2` Cross-validated R^2

`accuracy` standard accuracy that is only relevant if this is a classification problem

`num_OPLS_fact` number of orthogonal components

`permutation_Q2s` Q2 from each permutation test

`pvalue` p-value using the permutation test

Examples

```
res <- run_opls(rand(10,10),rand(10,1),100,-1)
```

Index