Package ‘TPLSr’

December 19, 2025

Type Package

Title Thresholded Partial Least Squares Model for Neuroimaging Data

Version 1.0.5

Description Uses thresholded partial least squares algorithm to create a regression or classification model. For more information, see Lee, Bradlow, and Kable <doi:10.1016/j.crmeth.2022.100227>.

License GPL-3

Depends R (>= 3.5), plotly (>= 4.9.2.1)

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.3.3

LazyDataCompression xz

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Sangil Lee [aut, cre]

Maintainer Sangil Lee <sangillee3rd@gmail.com>

Repository CRAN

Date/Publication 2025-12-19 05:40:07 UTC

Contents

`evalTuningParam`	*Evaluating cross-validation performance of a TPLS_cv model at compvec and threshvec*

Description

Evaluating cross-validation performance of a TPLS_cv model at compvec and threshvec

Usage

```
evalTuningParam(
  TPLScvmdl,
  type = c("Pearson", "negMSE", "ACC", "AUC", "LLbinary", "Spearman"),
  X,
  Y,
  compvec,
  threshvec,
  subfold = NULL
)
```

Arguments

`TPLScvmdl`	TPLS_cv model created from `TPLS_cv`
`type`	CV performance metric type. One of LLbinary, negMSE, Pearson, Spearman, AUC, ACC.
`X`	The same X as used in `TPLScvmdl`.
`Y`	The SAME Y as used in `TPLScvmdl`.
`compvec`	Vector of number of components to test in cross-validation.
`threshvec`	Vector of threshold level (0 ~ 1) to test in cross-validation.
`subfold`	(Optional) vector of subdivision within testing fold to calculate performance. For example scan run division within subject.

Value

A evalTuningParam object that contains the following attributes.

- `type`: Cross validation performance measure type, as specified in the input
- `threshval`: Same as the input threshvec
- `compval`: Same as the input compvec
- `perfmat`: Performance measure 3D matrix: length(compvec)-by-length(threshvec)-by-numfold
- `perf_best`: Best CV performance out of all combinations of compvec and threshvec
- `compval_best`: Number of components that gave the best performance (i.e., perf_best)
- `threshval_best`: Threshold level that gave the best performance (i.e., perf_best)

- `perf_1se` : Performance of the most parsimonious model (least number of coefficients) that is within 1 standard error of perf_best.
- `compval_1se` : Number of components that gave perf_1se
- `threshval_1se` : Threshold level that gave perf_1se
- `best_at_threshold` : a 3-column matrix; first column is max performance at threshold, second column is threshold values, third column is number of components for the best model at threshold

---

`makePredictor` *Method for extracting the T-PLS predictor at a given compval and threshval*

Description

Method for extracting the T-PLS predictor at a given compval and threshval

Usage

```
makePredictor(TPLSmdl, compval, threshval)
```

Arguments

`TPLSmdl`	A TPLS object created from using function `TPLS`
`compval`	Vector of number of components to use in final predictor. Providing a vector will provide multiple betamaps (e.g., c(3,4,5) will provide three betamaps each with 3, 4, and 5 PLS components)
`threshval`	Threshold number between 0 and 1 (inclusive) for thresholding the betamap. This must be a scalar.

Value

- `bias`: The intercept of the extracted model. Vector of intercepts if compval is a vector.
- `betamap`: Column vector of betamap. Matrix of betamaps if compval is a vector.

`plotTuningSurface` *Plots the tuning surface of TPLS*

Description

Plots the tuning surface of TPLS

Usage

```
plotTuningSurface(object)
```

Arguments

`object`	: evalTuningParam object

`TPLS` *Constructor method for fitting a T-PLS model with given data X and Y.*

Description

Constructor method for fitting a T-PLS model with given data X and Y.

Usage

```
TPLS(X, Y, NComp = 25, W = NULL, nmc = 0)
```

Arguments

`X`	Numerical matrix of predictors. Typically single-trial betas where each column is a voxel and row is observation
`Y`	Variable to predict. Binary 0 and 1 in case of classification, continuous variable in case of regression
`NComp`	(Optional) Number of PLS components to compute. Default is 25.
`W`	(Optional) Observation weights. By default, all observations have equal weight.
`nmc`	(Optional) 'no mean centering'. Default is 0. If 1, T-PLS will skip mean-centering. This option is only provided in case you already mean-centered the data and want to save some memory usage.

Value

A TPLS object that contains the following attributes. Most of the time, you won't need to access the attributes.

- `NComp`: The number of components you specified in the input
- `W`: Normalized version of the observation weights (i.e., they sum to 1)
- `MtrainX`: Column mean of X. Weighted mean if W is given.
- `MtrainY`: Mean of Y. Weighted mean if W is given.
- `scoreCorr`: Correlation between Y and each PLS component. Weighted correlation if W is given.
- `pctVar`: Proportion of variance of Y that each component explains.
- `betamap`: v-by-NComp matrix of TPLS coefficients for each of the v variables, provided at each model with NComp components.
- `threshmap` : v-by-NComp matrix of TPLS threshold values (0~1) for each of the v variables, provided at each model with NComp components.

See vignettes for tutorial

`TPLSdat` *Sample participant data from a left-right button press task*

Description

A dataset containing five sample participant's binary button presses inside the scanner (left/right).

Usage

`TPLSdat`

Format

A data frame with following variables

X Brain image single trial coefficients. N-by-v matrix

Y Left = 0, Right = 1, binary indicator of participant choice

subj Subject number (i.e., 1, 2, 3)

run Run number (i.e., 1, 2, 3, 4, 5, 6, 7, 8)

mask Binary 3D brain image that indexes where the variables in X came from.

Source

Kable, J. W., Caulfield, M. K., Falcone, M., McConnell, M., Bernardo, L., Parthasarathi, T., ... & Diefenbach, P. (2017). No effect of commercial cognitive training on brain activity, choice behavior, or cognitive performance. Journal of Neuroscience, 37(31), 7390-7402.

TPLSpredict — *Method for making predictions on a testing dataset testX*

Description

Method for making predictions on a testing dataset testX

Usage

```
TPLSpredict(TPLSmdl, compval, threshval, testX)
```

Arguments

`TPLSmdl`	A TPLS object created from using function `TPLS`
`compval`	Vector of number of components to use in final predictor. Providing a vector will provide multiple predictions (e.g., c(3,4,5) will provide three prediction columns each with 3, 4, and 5 PLS components)
`threshval`	Threshold number between 0 and 1 (inclusive) for thresholding the betamap. This must be a scalar.
`testX`	Data that you want to predict the Y of

Value

- `score`: Column vector of prediction scores. Matrix of scores if compval is a vector.

TPLS_cv — *Constructor method for fitting a cross-validation T-PLS model*

Description

Constructor method for fitting a cross-validation T-PLS model

Usage

```
TPLS_cv(X, Y, CVfold, NComp = 25, W = NULL, nmc = 0)
```

Arguments

`X`	Numerical matrix of predictors. Typically single-trial betas where each column is a voxel and row is observation
`Y`	Variable to predict. Binary 0 and 1 in case of classification, continuous variable in case of regression

`CVfold`	Cross-validation testing fold information. Can either be a vector or a matrix, the latter being more general. Vector: n-by-1 vector. Each element is a number ranging from 1 ~ numfold to identify which testing fold each observation belongs to Matrix: n-by-numfold matrix. Each column indicates the testing data with 1 and training data as 0. Example: For leave-one-out CV, Vector would be 1:n, Matrix form would be eye(n) Matrix form is more general as it can have same trial be in multiple test folds
`NComp`	(Optional) Number of PLS components to compute. Default is 25.
`W`	(Optional) Observation weights. Optional input. By default, all observations have equal weight. Can either be a n-by-1 vector or a n-by-nfold matrix where each column is observation weights in that CV fold
`nmc`	(Optional) 'no mean centering'. See TPLS for more detail. Turning this on will skip mean centering on all cross validation folds, so they should all be mean-centered already

Value

A TPLS_cv object that contains the following attributes. Most of the time, you won't need to access the attributes.

- `NComp`: The number of components you specified in the input
- `numfold`: Total number of cross-validation folds
- `CVfold`: A matrix of indicators for testing data for each cross validation fold in each column
- `cvMdls` : A vector of TPLS models, one for each fold.

See vignettes for tutorial

Index