Package ‘TANDEM’

October 12, 2022

Type Package

Title A Two-Stage Approach to Maximize Interpretability of Drug Response Models Based on Multiple Molecular Data Types

Version 1.0.3

Date 2019-11-18

Author Nanne Aben

Maintainer Nanne Aben <nanne.aben@gmail.com>

Description A two-stage regression method that can be used when various input data types are correlated, for example gene expression and methylation in drug response prediction. In the first stage it uses the upstream features (such as methylation) to predict the response variable (such as drug response), and in the second stage it uses the downstream features (such as gene expression) to predict the residuals of the first stage. In our manuscript (Aben et al., 2016, <doi:10.1093/bioinformatics/btw449>), we show that using TANDEM prevents the model from being dominated by gene expression and that the features selected by TANDEM are more interpretable.

Depends R (>= 2.10)

Imports glmnet (>= 3.0), Matrix

License GPL-2

LazyData TRUE

RoxygenNote 7.0.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-11-25 15:30:02 UTC

R topics documented:

coef.tandem *Returns the regression coefficients from a TANDEM fit*

Description

Returns the regression coefficients from a TANDEM fit.

Usage

```
## S3 method for class 'tandem'
coef(object, ...)
```

Arguments

`object`	A tandem-object, as returned by tandem()
`...`	Not used. Other arguments for coef().

Value

The regression coefficients.

Examples

```
# unpack example data
x = example_data$x
y = example_data$y
upstream = example_data$upstream

# fit a tandem model, determine the coefficients and create a prediction
fit = tandem(x, y, upstream, alpha=0.5)
beta = coef(fit)
y_hat = predict(fit, newx=x)
```

`example_data` — *A small artificial data set*

Description

A small artificial data set

Usage

```
example_data
```

Format

A small artificial data set, containing 200 samples, 20 upstream features and 20 downstream features

x A 200 x 40 feature matrix

y A 200 x 1 response vector

upstream A 40 x 1 boolean vector indicating which features are upstream features

data_types A 40 x 1 vector indicating for each feature to which data type they belong

`nested.cv` — *Estimating predictive performance via nested cross-validation*

Description

Performs a nested cross-validation to assess the predictive performance. The inner loop is used to determine the optimal lambda (as in cv.glmnet) and the outer loop is used to asses the predictive performance in an unbiased way.

Usage

```
nested.cv(
  x,
  y,
  upstream,
  method = "tandem",
  family = "gaussian",
  nfolds = 10,
  nfolds_inner = 10,
  foldid = NULL,
  lambda_upstream = "lambda.1se",
  lambda_downstream = "lambda.1se",
  lambda_glmnet = "lambda.1se",
  ...
)
```

Arguments

`x`	A feature matrix, where the rows correspond to samples and the columns to features.
`y`	A vector containing the response.
`upstream`	A logical index vector that indicates for each feature whether it's upstream (TRUE) or downstream (FALSE).
`method`	Indicates whether the nested cross-validation is performed on TANDEM or on the classic approach (glmnet). Should be either "tandem" or "glmnet".
`family`	The family parameter that's passed to cv.glmnet(). Currently, only family='gaussian' is supported.
`nfolds`	Number of cross-validation folds (default is 10) used in the outer cross-validation loop.
`nfolds_inner`	Number of cross-validation folds (default is 10) used to determine the optimal lambda in the inner cross-validation loop.
`foldid`	An optional vector indicating in which cross-validation fold each sample should be in the outer cross-validation loop. Overrides nfolds when used.
`lambda_upstream`	Only used when method='tandem'. For the first stage (using the upstream features), should glmnet use lambda.min or lambda.1se? Default is lambda.1se.
`lambda_downstream`	Only used when method='tandem'. For the second stage (using the downstream features), should glmnet use lambda.min or lambda.1se? Default is lambda.1se.
`lambda_glmnet`	Only used when method='glmnet'. Should glmnet use lambda.min or lambda.1se? Default is lambda.1se.
`...`	Other parameters that are passed to cv.glmnet().

Value

The predicted response vector y_hat and the mean-squared error (MSE).

Examples

```
# unpack example data
x = example_data$x
y = example_data$y
upstream = example_data$upstream

# assess the prediction error in a nested cv-loop
# fix the seed to have the same foldids between the two methods
set.seed(1)
cv_tandem = nested.cv(x, y, upstream, method="tandem", alpha=0.5)
set.seed(1)
cv_glmnet = nested.cv(x, y, upstream, method="glmnet", alpha=0.5)
barplot(c(cv_tandem$mse, cv_glmnet$mse), ylab="MSE", names=c("TANDEM", "Classic approach"))
```

predict.tandem *Creates a prediction using a tandem-object*

Description

Creates a prediction using a tandem-object.

Usage

```
## S3 method for class 'tandem'
predict(object, newx, ...)
```

Arguments

`object`	A tandem-object, as returned by tandem()
`newx`	A feature matrix, where the rows correspond to samples and the columns to features.
`...`	Not used. Other arguments for predict().

Value

The predicted response vector.

Examples

```
# unpack example data
x = example_data$x
y = example_data$y
upstream = example_data$upstream

# fit a tandem model, determine the coefficients and create a prediction
fit = tandem(x, y, upstream, alpha=0.5)
beta = coef(fit)
y_hat = predict(fit, newx=x)
```

relative.contributions *Determine the relative contribution per data type*

Description

For each data type, determine its relative contribution to the overall prediction.

Usage

```
relative.contributions(fit, x, data_types, lambda_glmnet = "lambda.1se")
```

Arguments

`fit`	Either a tandem-object or a cv.glmnet-object
`x`	The feature matrix used to train the fit, where the rows correspond to samples and the columns to features.
`data_types`	A vector of the same length as the number of features, that indicates for each feature to which data type it belongs. This vector doesn't need to correspond to the 'upstream' vector used in tandem(). For example, the upstream features be spread across various data types (such as mutation, CNA, methylation and cancer type) and the downstream features could be gene expression.
`lambda_glmnet`	Only used when fit is a cv.glmnet object. Should glmnet use lambda.min or lambda.1se? Default is lambda.1se. Note that for TANDEM objects, the lambda_upstream and lambda_downstream parameters should be specified during the tandem() call, as they are used while fitting the model.

Value

A vector that indicates the relative contribution per data type. These numbers sum up to one.

Examples

```
## simple example
# unpack example data
x = example_data$x
y = example_data$y
upstream = example_data$upstream
data_types = example_data$data_types

# fit TANDEM model
fit = tandem(x, y, upstream, alpha=0.5)

# assess the relative contribution of upstream and downstream features
contr = relative.contributions(fit, x, data_types)
barplot(contr, ylab="Relative contribution", ylim=0:1)

## comparing TANDEM and classic model (glmnet)
# unpack example data
x = example_data$x
y = example_data$y
upstream = example_data$upstream
data_types = example_data$data_types

# fix the cv folds, to facilitate a comparison between models
set.seed(1)
n = nrow(x)
nfolds = 10
foldid = ceiling(sample(1:n)/n * nfolds)

# fit both a TANDEM and a classic model (glmnet)
fit = tandem(x, y, upstream, alpha=0.5)
library(glmnet)
```

```
fit2 = cv.glmnet(x, y, alpha=0.5, foldid=foldid)

# assess the relative contribution of upstream and downstream features
# using both methods
contr_tandem = relative.contributions(fit, x, data_types)
contr_glmnet = relative.contributions(fit2, x, data_types)
par(mfrow=c(1,2))
barplot(contr_tandem, ylab="Relative contribution", main="TANDEM", ylim=0:1)
barplot(contr_glmnet, ylab="Relative contribution", main="Classic approach", ylim=0:1)
par(mfrow=c(1,1))
```

tandem — *Fits a TANDEM model by performing a two-stage regression*

Description

Fits a TANDEM model by performing a two-stage regression. In the first stage, all upstream features (x[,upstream]) are regressed on the output y. In the second stage, the downstream features (x[,!upstream]) are regressed on the residuals of the first stage. In both stages Elastic Net regression (as implemented in cv.glmnet() from the glmnet package) is used to perform the regression.

Usage

```
tandem(
  x,
  y,
  upstream,
  family = "gaussian",
  nfolds = 10,
  foldid = NULL,
  lambda_upstream = "lambda.1se",
  lambda_downstream = "lambda.1se",
  ...
)
```

Arguments

`x`	A feature matrix, where the rows correspond to samples and the columns to features.
`y`	A vector containing the response.
`upstream`	A boolean vector that indicates for each feature whether it's upstream (TRUE) or downstream (FALSE).
`family`	The family parameter that's passed to cv.glmnet(). Currently, only family='gaussian' is supported.
`nfolds`	Number of cross-validation folds (default is 10) used to determine the optimal lambda in cv.glmnet().

`foldid`	An optional vector indicating in which cross-validation fold each sample should be. Overrides nfolds when used.
`lambda_upstream`	For the first stage (using the upstream features), should glmnet use lambda.min or lambda.1se? Default is lambda.1se.
`lambda_downstream`	For the second stage (using the downstream features), should glmnet use lambda.min or lambda.1se? Default is lambda.1se.
`...`	Other parameters that are passed to cv.glmnet().

Value

A tandem-object.

Examples

```
# unpack example data
x = example_data$x
y = example_data$y
upstream = example_data$upstream

# fit a tandem model, determine the coefficients and create a prediction
fit = tandem(x, y, upstream, alpha=0.5)
beta = coef(fit)
y_hat = predict(fit, newx=x)
```

Index