Package 'TSPC'

July 6, 2015

Type Package

Title Prediction using time-course gene expression

Version 2.0

Date 2014-01-17

Author Yuping Zhang

Maintainer Yuping Zhang <yuping.zhang@uconn.edu>

Description Performs survival and quantitative outcome using
time-course gene expression, described in the following
papers:
Zhang Y, Tibshirani RJ, Davis RW. Predicting patient survival from longitudinal gene expression. Stat Appl Genet Mol Biol. 2010;9(1):Article41. Epub 2010 Nov 22.
Zhang Y, Ouyang Z. Predicting quantitative outcomes of patients using longitudinal gene expression. Sri Lankan Journal of Applied Statistics, 5(4), 117-126.

License GPL (>= 2)

LazyLoad TRUE

Depends superpc, survival

Repository CRAN

Date/Publication 2012-10-29 08:57:45

R topics documented:

TSPC-package *TSPC*

Description

Performs survival and quantitative outcome using time-course gene expression, described in the following papers: Zhang Y, Tibshirani RJ, Davis RW. Predicting patient survival from longitudinal gene expression. Stat Appl Genet Mol Biol. 2010;9(1):Article41. Epub 2010 Nov 22.\ Zhang Y, Ouyang Z. Predicting quantitative outcomes of patients using longitudinal gene expression. Sri Lankan Journal of Applied Statistics, 5(4), 117-126.

Details

Package:	TSPC
Type:	Package
Version:	2.0
Date:	2014-11-25
License:	GPL
LazyLoad:	yes

Author(s)

Yuping Zhang <yupingz@stanford.edu>

References

Zhang Y, Tibshirani RJ, Davis RW. Predicting patient survival from longitudinal gene expression. Stat Appl Genet Mol Biol. 2010;9(1):Article41. Epub 2010 Nov 22.

tspc.cv *Cross-validation*

Description

This function uses a form of cross-validation to estimate the optimal feature threshold in supervised principal components

Usage

```
tspc.cv(fit, data, seed = 123, topfea = TRUE, n.topfea = 1000, n.threshold = 20, n.fold = NULL, fol
```

Arguments

fit	Object returned by tspc.train
data	Data object of form described in tspc.train documentation
seed	A Numeric number
topfea	If it is TRUE, then the tuning paparmeter is the number of features
n.topfea	Maximum number of features used as the tuning parameter
n.threshold	Number of thresholds, when using the number of thresholds as a tuning parameter
n.fold	Number of cross-validation folds
folds	Lists of indices of cross-validation folds (optional)
n.components	Number of cross-validation components to use: 1,2 or 3.
min.features	Minimum number of features to include, in determining range for threshold. Default 5.
max.features	Maximum number of features to include, in determining range for threshold. Default is total number of features in the dataset.
type	"survival" or "regression"

Details

This function uses a form of cross-validation to estimate the optimal feature threshold.

Value

list(thresholds = thresholds, n.threshold = n.threshold, nonzero = nonzero, scor = scor, scor.lower = scor.lower, scor.upper = scor.upper, folds = folds, n.fold = n.fold, featurescores.folds = featurescores.folds, type = type)

thresholds	Vector of thresholds considered
n.threshold	Number of thresholds
nonzero	Number of features exceeding each value of the threshold
scor	Full CV scores
scor.lower	Full CV scores minus one standard error of scores
scor.upper	Full CV scores plus one standard error of scores
folds	Indices of CV folds used
n.fold	Number of folds used in the cross-validation
featurescores.folds	Feature scores for each fold
type	problem type

Author(s)

Yuping Zhang

Examples

```
x = list()
for(i in 1:2){
set.seed(i+123)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep="")))

x = list()
for(i in 1:2){
set.seed(i+133)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data.test = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep

fit = tspc.train(data, data.test, type="survival")

cv.obj = tspc.cv(fit$fit.obj, data, type="survival", n.fold=2)
```

---

tspc.plotcv *Plot output from tspc.cv*

Description

Plots pre-validation results from plotcv, to aid in choosing best threshold

Usage

```
tspc.plotcv(object)
```

Arguments

object	Object returned by tspc.cv

Author(s)

Yuping Zhang

Examples

```
x = list()
for(i in 1:2){
set.seed(i+123)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)
```

```
data = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep="")))

x = list()
for(i in 1:2){
set.seed(i+133)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data.test = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep

fit = tspc.train(data, data.test, type="survival")

cv.obj = tspc.cv(fit$fit.obj, data, type="survival", n.fold=2)
tspc.plotcv(cv.obj)
```

`tspc.predict` *Form principal components predictor from a trained tspc object*

Description

Computes supervised principal components, using scores from "object"

Usage

```
tspc.predict(object, data, newdata, threshold, n.components = 3, prediction.type = c("continuous",
```

Arguments

`object`	Object fit.obj returned by tspc.train
`data`	List of projection of training data returned by tspc.train, object proj.obj$wdata.train
`newdata`	List of projection of test data returned by tspc.train, object proj.obj$wdata.test
`threshold`	Threshold for scores.
`n.components`	Number of principal components to compute. Should be 1,2 or 3.
`prediction.type`	"continuous" for raw principal component(s); "discrete" for principal component categorized in equal bins; "nonzero" for indices of features that pass the threshold
`n.class`	Number of classes into which predictor is binned (for prediction.type="discrete"

Value

list(v.pred = out, u = x.sml.svd$u, d = x.sml.svd$d, which.features = which.features, v.pred.1df = v.pred.1df, n.components = n.pc, coef = result$coef, call = this.call, prediction.type = prediction.type)

`v.pred`	Supervised principal componients predictor
`u`	U matrix from svd of weighted feature matrix
`d`	singual values from svd of weighted feature matrix
`which.features`	Indices of features exceeding threshold
`n.components`	Number of supervised principal components requested

Author(s)

Yuping Zhang

Examples

```
x = list()
for(i in 1:2){
set.seed(i+123)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep="")))

x = list()
for(i in 1:2){
set.seed(i+133)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data.test = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep

fit = tspc.train(data, data.test, type="survival")

predict.obj<- tspc.predict(fit$fit.obj, fit$proj.obj$data.train, fit$proj.obj$data.test, threshold=1.0, n.co
```

`tspc.project` *Project time-course gene expression to weighted gene expression*

Description

Project time-course gene expression to weighted gene expression

Usage

```
tspc.project(data, data.test, type = c("survival", "regression"))
```

Arguments

`data`	List of training data, of form described in tspc.train documentation
`data.test`	List of test data, of form described in tspc.test documentation
`type`	Problem type: "survival" for censored survival outcome, or "regression" for simple quantitative outcome.

Value

list(data.train = wdata.train, data.test = wdata.test)

`data.train`	Projection of training data
`data.test`	Projection of test data

Author(s)

Yuping Zhang

`tspc.train` *Prediction using time-course gene expression*

Description

Does prediction of a quantitative regression or survival outcome, using the time-course gene expression.

Usage

```
tspc.train(data, data.test, type = c("survival", "regression"), s0.perc = 0.5)
```

Arguments

`data`	Data object with components x- a list of p by n matrix of features, one observation per column, one matrix per time point; y- n-vector of outcome measurements; censoring.status- n-vector of censoring censoring.status (1= died or event occurred, 0=survived, or event was censored), needed for a censored survival outcome.
`data.test`	Data object with components x- a list of p by n matrix of features, one observation per column, one matrix per time point; y- n-vector of outcome measurements; censoring.status- n-vector of censoring censoring.status (1= died or event occurred, 0=survived, or event was censored), needed for a censored survival outcome.
`type`	Problem type: "survival" for censored survival outcome, or "regression" for simple quantitative outcome.
`s0.perc`	Factor for denominator of score statistic, between 0 and 1: the percentile of standard deviation values added to the denominator. Default is 0.5 (the median)

Value

`proj.obj`	projection of training data and test data
`fit.obj`	fitted object using training data

Author(s)

Yuping Zhang

References

Zhang Y, Tibshirani RJ, Davis RW. Predicting patient survival from longitudinal gene expression. Stat Appl Genet Mol Biol. 2010;9(1):Article41. Epub 2010 Nov 22.

Examples

```
x = list()
for(i in 1:2){
set.seed(i+123)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep="")))

x = list()
for(i in 1:2){
set.seed(i+133)
x[[i]] = matrix(rnorm(500*100), ncol=100)
}
y = sample(c(5:100), size=100, replace=TRUE)
censoring = sample(c(0,1), size=100, replace=TRUE)

data.test = list(x = x, y=y, censoring.status=censoring, genenames = as.character(paste("gene", c(1:500), sep

obj = tspc.train(data, data.test, type="survival")
```

Index