Package 'nnGarrote'

October 13, 2022

Type Package

Title Non-Negative Garrote Estimation with Penalized Initial Estimators

Version 1.0.4

Date 2021-10-05

Author Anthony Christidis <anthony.christidis@stat.ubc.ca>, Stefan Van Aelst <stefan.vanaelst@kuleuven.be>, Ruben Zamar <ruben@stat.ubc.ca>

Maintainer Anthony Christidis <anthony.christidis@stat.ubc.ca>

Description Functions to compute the non-negative garrote estimator as proposed by Breiman (1995) <https://www.jstor.org/stable/1269730> with the penalized initial estimators extension as proposed by Yuan and Lin (2007) <https://www.jstor.org/stable/4623260>.

License GPL (>= 2)

Biarch true

Imports glmnet

RoxygenNote 7.1.1

Suggests testthat, mvnfast

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-07 07:40:19 UTC

R topics documented:

`coef.cv.nnGarrote` *Coefficients for cv.nnGarrote Object*

Description

`coef.cv.nnGarrote` returns the coefficients for a cv.nnGarrote object.

Usage

```
## S3 method for class 'cv.nnGarrote'
coef(object, optimal.only = TRUE, ...)
```

Arguments

`object`	An object of class cv.nnGarrote
`optimal.only`	A boolean variable (TRUE default) to indicate if only the coefficient of the optimal split are returned.
`...`	Additional arguments for compatibility.

Value

A matrix with the coefficients of the `cv.nnGarrote` object.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

`cv.nnGarrote`

Examples

```
# Setting the parameters
p <- 500
n <- 100
n.test <- 5000
sparsity <- 0.15
rho <- 0.5
SNR <- 3
set.seed(0)
# Generating the coefficient
p.active <- floor(p*sparsity)
a <- 4*log(n)/sqrt(n)
neg.prob <- 0.2
nonzero.betas <- (-1)^(rbinom(p.active, 1, neg.prob))*(a + abs(rnorm(p.active)))
true.beta <- c(nonzero.betas, rep(0, p-p.active))
```

```
# Two groups correlation structure
Sigma.rho <- matrix(0, p, p)
Sigma.rho[1:p.active, 1:p.active] <- rho
diag(Sigma.rho) <- 1
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma.rho %*% true.beta)/SNR))

# Simulate some data
library(mvnfast)
x.train <- mvnfast::rmvn(n, mu=rep(0,p), sigma=Sigma.rho)
y.train <- 1 + x.train %*% true.beta + rnorm(n=n, mean=0, sd=sigma.epsilon)
x.test <- mvnfast::rmvn(n.test, mu=rep(0,p), sigma=Sigma.rho)
y.test <- 1 + x.test %*% true.beta + rnorm(n.test, sd=sigma.epsilon)

# Applying the NNG with Ridge as an initial estimator
nng.out <- cv.nnGarrote(x.train, y.train, intercept=TRUE,
                        initial.model=c("LS", "glmnet")[2],
                        lambda.nng=NULL, lambda.initial=NULL, alpha=0,
                        nfolds=5)
nng.predictions <- predict(nng.out, newx=x.test)
mean((nng.predictions-y.test)^2)/sigma.epsilon^2
coef(nng.out)
```

`coef.nnGarrote` *Coefficients for nnGarrote Object*

Description

`coef.nnGarrote` returns the coefficients for a nnGarrote object.

Usage

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

Arguments

`object`	An object of class nnGarrote.
`...`	Additional arguments for compatibility.

Value

A matrix with the coefficients of the `nnGarrote` object.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

nnGarrote

Examples

```
# Setting the parameters
p <- 500
n <- 100
n.test <- 5000
sparsity <- 0.15
rho <- 0.5
SNR <- 3
set.seed(0)
# Generating the coefficient
p.active <- floor(p*sparsity)
a <- 4*log(n)/sqrt(n)
neg.prob <- 0.2
nonzero.betas <- (-1)^(rbinom(p.active, 1, neg.prob))*(a + abs(rnorm(p.active)))
true.beta <- c(nonzero.betas, rep(0, p-p.active))
# Two groups correlation structure
Sigma.rho <- matrix(0, p, p)
Sigma.rho[1:p.active, 1:p.active] <- rho
diag(Sigma.rho) <- 1
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma.rho %*% true.beta)/SNR))

# Simulate some data
library(mvnfast)
x.train <- mvnfast::rmvn(n, mu=rep(0,p), sigma=Sigma.rho)
y.train <- 1 + x.train %*% true.beta + rnorm(n=n, mean=0, sd=sigma.epsilon)
x.test <- mvnfast::rmvn(n.test, mu=rep(0,p), sigma=Sigma.rho)
y.test <- 1 + x.test %*% true.beta + rnorm(n.test, sd=sigma.epsilon)

# Applying the NNG with Ridge as an initial estimator
nng.out <- nnGarrote(x.train, y.train, intercept=TRUE,
                     initial.model=c("LS", "glmnet")[2],
                     lambda.nng=NULL, lambda.initial=NULL, alpha=0)
nng.predictions <- predict(nng.out, newx=x.test)
nng.coef <- coef(nng.out)
```

---

cv.nnGarrote *Non-negative Garrote Estimator - Cross-Validation*

Description

cv.nnGarrote computes the non-negative garrote estimator with cross-validation.

Usage

```
cv.nnGarrote(
  x,
  y,
  intercept = TRUE,
  initial.model = c("LS", "glmnet")[1],
  lambda.nng = NULL,
  lambda.initial = NULL,
  alpha = 0,
  nfolds = 5,
  verbose = TRUE
)
```

Arguments

x	Design matrix.
y	Response vector.
intercept	Boolean variable to determine if there is intercept (default is TRUE) or not.
initial.model	Model used for the groups. Must be one of "LS" (default) or "glmnet".
lambda.nng	Shinkage parameter for the non-negative garrote. If NULL(default), it will be computed based on data.
lambda.initial	The shinkrage parameter for the "glmnet" regularization.
alpha	Elastic net mixing parameter for initial estimate. Should be between 0 (default) and 1.
nfolds	Number of folds for the cross-validation procedure.
verbose	Boolean variable to determine if console output for cross-validation progress is printed (default is TRUE).

Value

An object of class cv.nnGarrote

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

coef.cv.nnGarrote, predict.cv.nnGarrote

Examples

```
# Setting the parameters
p <- 500
n <- 100
n.test <- 5000
```

```
sparsity <- 0.15
rho <- 0.5
SNR <- 3
set.seed(0)
# Generating the coefficient
p.active <- floor(p*sparsity)
a <- 4*log(n)/sqrt(n)
neg.prob <- 0.2
nonzero.betas <- (-1)^(rbinom(p.active, 1, neg.prob))*(a + abs(rnorm(p.active)))
true.beta <- c(nonzero.betas, rep(0, p-p.active))
# Two groups correlation structure
Sigma.rho <- matrix(0, p, p)
Sigma.rho[1:p.active, 1:p.active] <- rho
diag(Sigma.rho) <- 1
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma.rho %*% true.beta)/SNR))

# Simulate some data
library(mvnfast)
x.train <- mvnfast::rmvn(n, mu=rep(0,p), sigma=Sigma.rho)
y.train <- 1 + x.train %*% true.beta + rnorm(n=n, mean=0, sd=sigma.epsilon)
x.test <- mvnfast::rmvn(n.test, mu=rep(0,p), sigma=Sigma.rho)
y.test <- 1 + x.test %*% true.beta + rnorm(n.test, sd=sigma.epsilon)

# Applying the NNG with Ridge as an initial estimator
nng.out <- cv.nnGarrote(x.train, y.train, intercept=TRUE,
                        initial.model=c("LS", "glmnet")[2],
                        lambda.nng=NULL, lambda.initial=NULL, alpha=0,
                        nfolds=5)
nng.predictions <- predict(nng.out, newx=x.test)
mean((nng.predictions-y.test)^2)/sigma.epsilon^2
coef(nng.out)
```

---

nnGarrote *Non-negative Garrote Estimator*

Description

nnGarrote computes the non-negative garrote estimator.

Usage

```
nnGarrote(
  x,
  y,
  intercept = TRUE,
  initial.model = c("LS", "glmnet")[1],
  lambda.nng = NULL,
  lambda.initial = NULL,
```

```
  alpha = 0
)
```

Arguments

x	Design matrix.
y	Response vector.
intercept	Boolean variable to determine if there is intercept (default is TRUE) or not.
initial.model	Model used for the groups. Must be one of "LS" (default) or "glmnet".
lambda.nng	Shinkage parameter for the non-negative garrote. If NULL(default), it will be computed based on data.
lambda.initial	The shinkrage parameter for the "glmnet" regularization. If NULL (default), optimal value is chosen by cross-validation.
alpha	Elastic net mixing parameter for initial estimate. Should be between 0 (default) and 1.

Value

An object of class nnGarrote.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

coef.nnGarrote, predict.nnGarrote

Examples

```
# Setting the parameters
p <- 500
n <- 100
n.test <- 5000
sparsity <- 0.15
rho <- 0.5
SNR <- 3
set.seed(0)
# Generating the coefficient
p.active <- floor(p*sparsity)
a <- 4*log(n)/sqrt(n)
neg.prob <- 0.2
nonzero.betas <- (-1)^(rbinom(p.active, 1, neg.prob))*(a + abs(rnorm(p.active)))
true.beta <- c(nonzero.betas, rep(0, p-p.active))
# Two groups correlation structure
Sigma.rho <- matrix(0, p, p)
Sigma.rho[1:p.active, 1:p.active] <- rho
diag(Sigma.rho) <- 1
```

```
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma.rho %*% true.beta)/SNR))

# Simulate some data
library(mvnfast)
x.train <- mvnfast::rmvn(n, mu=rep(0,p), sigma=Sigma.rho)
y.train <- 1 + x.train %*% true.beta + rnorm(n=n, mean=0, sd=sigma.epsilon)
x.test <- mvnfast::rmvn(n.test, mu=rep(0,p), sigma=Sigma.rho)
y.test <- 1 + x.test %*% true.beta + rnorm(n.test, sd=sigma.epsilon)

# Applying the NNG with Ridge as an initial estimator
nng.out <- nnGarrote(x.train, y.train, intercept=TRUE,
                     initial.model=c("LS", "glmnet")[2],
                     lambda.nng=NULL, lambda.initial=NULL, alpha=0)
nng.predictions <- predict(nng.out, newx=x.test)
nng.coef <- coef(nng.out)
```

`predict.cv.nnGarrote` *Predictions for cv.nnGarrote Object*

Description

`predict.cv.nnGarrote` returns the prediction for cv.nnGarrote for new data.

Usage

```
## S3 method for class 'cv.nnGarrote'
predict(object, newx, optimal.only = TRUE, ...)
```

Arguments

`object`	An object of class cv.nnGarrote
`newx`	A matrix with the new data.
`optimal.only`	A boolean variable (TRUE default) to indicate if only the coefficient of the optimal split are returned.
`...`	Additional arguments for compatibility.

Value

A matrix with the predictions of the `cv.nnGarrote` object.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

`cv.nnGarrote`

Examples

```
# Setting the parameters
p <- 500
n <- 100
n.test <- 5000
sparsity <- 0.15
rho <- 0.5
SNR <- 3
set.seed(0)
# Generating the coefficient
p.active <- floor(p*sparsity)
a <- 4*log(n)/sqrt(n)
neg.prob <- 0.2
nonzero.betas <- (-1)^(rbinom(p.active, 1, neg.prob))*(a + abs(rnorm(p.active)))
true.beta <- c(nonzero.betas, rep(0, p-p.active))
# Two groups correlation structure
Sigma.rho <- matrix(0, p, p)
Sigma.rho[1:p.active, 1:p.active] <- rho
diag(Sigma.rho) <- 1
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma.rho %*% true.beta)/SNR))

# Simulate some data
library(mvnfast)
x.train <- mvnfast::rmvn(n, mu=rep(0,p), sigma=Sigma.rho)
y.train <- 1 + x.train %*% true.beta + rnorm(n=n, mean=0, sd=sigma.epsilon)
x.test <- mvnfast::rmvn(n.test, mu=rep(0,p), sigma=Sigma.rho)
y.test <- 1 + x.test %*% true.beta + rnorm(n.test, sd=sigma.epsilon)

# Applying the NNG with Ridge as an initial estimator
nng.out <- cv.nnGarrote(x.train, y.train, intercept=TRUE,
                        initial.model=c("LS", "glmnet")[2],
                        lambda.nng=NULL, lambda.initial=NULL, alpha=0,
                        nfolds=5)
nng.predictions <- predict(nng.out, newx=x.test)
mean((nng.predictions-y.test)^2)/sigma.epsilon^2
coef(nng.out)
```

`predict.nnGarrote` *Predictions for nnGarrote Object*

Description

`predict.nnGarrote` returns the prediction for nnGarrote for new data.

Usage

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

Arguments

object	An object of class nnGarrote
newx	A matrix with the new data.
...	Additional arguments for compatibility.

Value

A matrix with the predictions of the nnGarrote object.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

nnGarrote

Examples

```
# Setting the parameters
p <- 500
n <- 100
n.test <- 5000
sparsity <- 0.15
rho <- 0.5
SNR <- 3
set.seed(0)
# Generating the coefficient
p.active <- floor(p*sparsity)
a <- 4*log(n)/sqrt(n)
neg.prob <- 0.2
nonzero.betas <- (-1)^(rbinom(p.active, 1, neg.prob))*(a + abs(rnorm(p.active)))
true.beta <- c(nonzero.betas, rep(0, p-p.active))
# Two groups correlation structure
Sigma.rho <- matrix(0, p, p)
Sigma.rho[1:p.active, 1:p.active] <- rho
diag(Sigma.rho) <- 1
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma.rho %*% true.beta)/SNR))

# Simulate some data
library(mvnfast)
x.train <- mvnfast::rmvn(n, mu=rep(0,p), sigma=Sigma.rho)
y.train <- 1 + x.train %*% true.beta + rnorm(n=n, mean=0, sd=sigma.epsilon)
x.test <- mvnfast::rmvn(n.test, mu=rep(0,p), sigma=Sigma.rho)
y.test <- 1 + x.test %*% true.beta + rnorm(n.test, sd=sigma.epsilon)

# Applying the NNG with Ridge as an initial estimator
nng.out <- nnGarrote(x.train, y.train, intercept=TRUE,
                     initial.model=c("LS", "glmnet")[2],
                     lambda.nng=NULL, lambda.initial=NULL, alpha=0)
```

```
nng.predictions <- predict(nng.out, newx=x.test)
nng.coef <- coef(nng.out)
```

Index