Package: stepSplitReg (via r-universe)

October 5, 2026

Type Package

Title Stepwise Split Regularized Regression

Version 1.0.5

Date 2025-03-30

Description Functions to perform stepwise split regularized regression. The approach first uses a stepwise algorithm to split the variables into the models with a goodness of fit criterion, and then regularization is applied to each model. The weights of the models in the ensemble are determined based on a criterion selected by the user.

License GPL (>= 2)

Biarch true

Imports Rcpp (>= 1.0.7), SplitGLM, nnls

Suggests testthat, mvnfast, vctrs

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

NeedsCompilation yes

Author Anthony Christidis [aut, cre], Stefan Van Aelst [aut], Ruben Zamar [aut]

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

Repository https://cran.r-universe.dev

Date/Publication 2025-03-30 17:32:00 UTC

RemoteUrl https://github.com/cran/stepSplitReg

RemoteRef HEAD

RemoteSha 8eb76735d15f5dc70e1d3859e5f35397be5a0f04

Contents

coef.cv.stepSplitReg 2
coef.stepSplitReg 3

cv.stepSplitReg 5
predict.cv.stepSplitReg 7
predict.stepSplitReg 9
stepSplitReg 10

Index **14**

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

Description

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

Usage

```
## S3 method for class 'cv.stepSplitReg'
coef(object, group_index = NULL, ...)
```

Arguments

`object`	An object of class cv.stepSplitReg
`group_index`	Groups included in the ensemble. Default setting includes all the groups.
`...`	Additional arguments for compatibility.

Value

The coefficients for the cv.stepSplitReg object.

Author(s)

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

See Also

`cv.stepSplitReg`

Examples

```
# Required Libraries
library(mvnfast)

# Setting the parameters
p <- 100
n <- 30
n.test <- 500
sparsity <- 0.2
rho <- 0.5
SNR <- 3
```

```
# 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)))

# Correlation structure
Sigma <- matrix(0, p, p)
Sigma[1:p.active, 1:p.active] <- rho
diag(Sigma) <- 1
true.beta <- c(nonzero.betas, rep(0 , p - p.active))

# Computing the noise parameter for target SNR
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma %*% true.beta)/SNR))

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

# Stepwise Split Regularized Regression
step.out <- cv.stepSplitReg(x.train, y.train, n_models = c(2, 3), max_variables = NULL, keep = 4/4,
                            model_criterion = c("F-test", "RSS")[1],
                           stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
                            stop_parameter = 0.05,
                            shrinkage = TRUE, alpha = 4/4, include_intercept = TRUE,
                           n_lambda = 50, tolerance = 1e-2, max_iter = 1e5, n_folds = 5,
                            model_weights = c("Equal", "Proportional", "Stacking")[1],
                            n_threads = 1)
step.coefficients <- coef(step.out, group_index = 1:step.out$n_models_optimal)
step.predictions <- predict(step.out, x.test, group_index = 1:step.out$n_models_optimal)
mspe.step <- mean((step.predictions-y.test)^2)/sigma.epsilon^2
```

`coef.stepSplitReg` *Coefficients for stepSplitReg Object*

Description

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

Usage

```
## S3 method for class 'stepSplitReg'
coef(object, group_index = NULL, ...)
```

Arguments

object	An object of class stepSplitReg
group_index	Groups included in the ensemble. Default setting includes all the groups.
...	Additional arguments for compatibility.

Value

The coefficients for the stepSplitReg object.

Author(s)

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

See Also

stepSplitReg

Examples

```
# Required Libraries
library(mvnfast)

# Setting the parameters
p <- 100
n <- 30
n.test <- 1000
sparsity <- 0.2
rho <- 0.5
SNR <- 3

# 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)))

# Correlation structure
Sigma <- matrix(0, p, p)
Sigma[1:p.active, 1:p.active] <- rho
diag(Sigma) <- 1
true.beta <- c(nonzero.betas, rep(0 , p - p.active))

# Computing the noise parameter for target SNR
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma %*% true.beta)/SNR))

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

```
# Stepwise Split Regularized Regression
step.out <- stepSplitReg(x.train, y.train, n_models = 3, max_variables = NULL, keep = 4/4,
                         model_criterion = c("F-test", "RSS")[1],
                         stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
                         stop_parameter = 0.05,
                         shrinkage = TRUE, alpha = 4/4, include_intercept = TRUE,
                         n_lambda = 50, tolerance = 1e-2, max_iter = 1e5, n_folds = 5,
                         model_weights = c("Equal", "Proportional", "Stacking")[1])
step.coefficients <- coef(step.out, group_index = 1:step.out$n_models)
step.predictions <- predict(step.out, x.test, group_index = 1:step.out$n_models)
mspe.step <- mean((step.predictions-y.test)^2)/sigma.epsilon^2
```

`cv.stepSplitReg` *Cross Validation - Stepwise Split Regularized Regression*

Description

`cv.stepSplitReg` performs the CV procedure for stepwise split regularized regression.

Usage

```
cv.stepSplitReg(
  x,
  y,
  n_models = NULL,
  max_variables = NULL,
  keep = 1,
  model_criterion = c("F-test", "RSS")[1],
  stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
  stop_parameter = 0.05,
  shrinkage = TRUE,
  alpha = 3/4,
  include_intercept = TRUE,
  n_lambda = 100,
  tolerance = 0.001,
  max_iter = 1e+05,
  n_folds = 10,
  model_weights = c("Equal", "Proportional", "Stacking")[1],
  n_threads = 1
)
```

Arguments

`x`	Design matrix.
`y`	Response vector.
`n_models`	Number of models into which the variables are split.

max_variables	Maximum number of variables that a model can contain.
keep	Proportion of models to keep based on their individual cross-validated errors. Default is 1.
model_criterion	Criterion for adding a variable to a model. Must be one of c("F-test", "RSS"). Default is "F-test".
stop_criterion	Criterion for determining when a model is saturated. Must be one of c("F-test", "pR2", "aR2", "R2", "Fixed"). Default is "F-test".
stop_parameter	Parameter value for the stopping criterion. Default is 0.05 for "F-test".
shrinkage	TRUE or FALSE parameter for shrinkage of the final models. Default is TRUE.
alpha	Elastic net mixing parmeter for model shrinkage. Default is 3/4.
include_intercept	TRUE or FALSE parameter for the inclusion of an intercept term.
n_lambda	Number of candidates for the sparsity penalty parameter. Default is 100.
tolerance	Convergence criteria for the coefficients. Default is 1e-3.
max_iter	Maximum number of iterations in the algorithm. Default is 1e5.
n_folds	Number of cross-validation folds. Default is 10.
model_weights	Criterion to determine the weights of the model for prediciton. Must be one of c("Equal", "Proportional", "Stacking"). Default is "Equal".
n_threads	Number of threads. Default is 1.

Value

An object of class cv.stepSplitReg.

Author(s)

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

See Also

coef.cv.stepSplitReg, predict.cv.stepSplitReg

Examples

```
# Required Libraries
library(mvnfast)

# Setting the parameters
p <- 100
n <- 30
n.test <- 500
sparsity <- 0.2
rho <- 0.5
SNR <- 3

# 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)))

# Correlation structure
Sigma <- matrix(0, p, p)
Sigma[1:p.active, 1:p.active] <- rho
diag(Sigma) <- 1
true.beta <- c(nonzero.betas, rep(0 , p - p.active))

# Computing the noise parameter for target SNR
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma %*% true.beta)/SNR))

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

# Stepwise Split Regularized Regression
step.out <- cv.stepSplitReg(x.train, y.train, n_models = c(2, 3), max_variables = NULL, keep = 4/4,
                            model_criterion = c("F-test", "RSS")[1],
                           stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
                            stop_parameter = 0.05,
                            shrinkage = TRUE, alpha = 4/4, include_intercept = TRUE,
                           n_lambda = 50, tolerance = 1e-2, max_iter = 1e5, n_folds = 5,
                            model_weights = c("Equal", "Proportional", "Stacking")[1],
                            n_threads = 1)
step.coefficients <- coef(step.out, group_index = 1:step.out$n_models_optimal)
step.predictions <- predict(step.out, x.test, group_index = 1:step.out$n_models_optimal)
mspe.step <- mean((step.predictions-y.test)^2)/sigma.epsilon^2
```

`predict.cv.stepSplitReg`

Predictions for cv.stepSplitReg Object

Description

`predict.cv.stepSplitReg` returns the predictions for a cv.stepSplitReg object.

Usage

```
## S3 method for class 'cv.stepSplitReg'
predict(object, newx, group_index = group_index, ...)
```

Arguments

object	An object of class cv.stepSplitReg
newx	New data for predictions.
group_index	Groups included in the ensemble. Default setting includes all the groups.
...	Additional arguments for compatibility.

Value

The predictions for the cv.stepSplitReg object.

Author(s)

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

See Also

cv.stepSplitReg

Examples

```
# Required Libraries
library(mvnfast)

# Setting the parameters
p <- 100
n <- 30
n.test <- 500
sparsity <- 0.2
rho <- 0.5
SNR <- 3

# 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)))

# Correlation structure
Sigma <- matrix(0, p, p)
Sigma[1:p.active, 1:p.active] <- rho
diag(Sigma) <- 1
true.beta <- c(nonzero.betas, rep(0 , p - p.active))

# Computing the noise parameter for target SNR
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma %*% true.beta)/SNR))

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

# Stepwise Split Regularized Regression
step.out <- cv.stepSplitReg(x.train, y.train, n_models = c(2, 3), max_variables = NULL, keep = 4/4,
                            model_criterion = c("F-test", "RSS")[1],
                           stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
                            stop_parameter = 0.05,
                            shrinkage = TRUE, alpha = 4/4, include_intercept = TRUE,
                           n_lambda = 50, tolerance = 1e-2, max_iter = 1e5, n_folds = 5,
                            model_weights = c("Equal", "Proportional", "Stacking")[1],
                            n_threads = 1)
step.coefficients <- coef(step.out, group_index = 1:step.out$n_models_optimal)
step.predictions <- predict(step.out, x.test, group_index = 1:step.out$n_models_optimal)
mspe.step <- mean((step.predictions-y.test)^2)/sigma.epsilon^2
```

`predict.stepSplitReg` *Predictions for stepSplitReg Object*

Description

`predict.stepSplitReg` returns the predictions for a stepSplitReg object.

Usage

```
## S3 method for class 'stepSplitReg'
predict(object, newx, group_index = NULL, ...)
```

Arguments

`object`	An object of class stepSplitReg
`newx`	New data for predictions.
`group_index`	Groups included in the ensemble. Default setting includes all the groups.
`...`	Additional arguments for compatibility.

Value

The predictions for the stepSplitReg object.

Author(s)

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

See Also

`stepSplitReg`

Examples

```
# Required Libraries
library(mvnfast)

# Setting the parameters
p <- 100
n <- 30
n.test <- 1000
sparsity <- 0.2
rho <- 0.5
SNR <- 3

# 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)))

# Correlation structure
Sigma <- matrix(0, p, p)
Sigma[1:p.active, 1:p.active] <- rho
diag(Sigma) <- 1
true.beta <- c(nonzero.betas, rep(0 , p - p.active))

# Computing the noise parameter for target SNR
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma %*% true.beta)/SNR))

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

# Stepwise Split Regularized Regression
step.out <- stepSplitReg(x.train, y.train, n_models = 3, max_variables = NULL, keep = 4/4,
                         model_criterion = c("F-test", "RSS")[1],
                         stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
                         stop_parameter = 0.05,
                         shrinkage = TRUE, alpha = 4/4, include_intercept = TRUE,
                         n_lambda = 50, tolerance = 1e-2, max_iter = 1e5, n_folds = 5,
                         model_weights = c("Equal", "Proportional", "Stacking")[1])
step.coefficients <- coef(step.out, group_index = 1:step.out$n_models)
step.predictions <- predict(step.out, x.test, group_index = 1:step.out$n_models)
mspe.step <- mean((step.predictions-y.test)^2)/sigma.epsilon^2
```

---

stepSplitReg *Stepwise Split Regularized Regression*

Description

stepSplitReg performs stepwise split regularized regression.

Usage

```
stepSplitReg(
  x,
  y,
  n_models = NULL,
  max_variables = NULL,
  keep = 1,
  model_criterion = c("F-test", "RSS")[1],
  stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
  stop_parameter = 0.05,
  shrinkage = TRUE,
  alpha = 3/4,
  include_intercept = TRUE,
  n_lambda = 100,
  tolerance = 0.001,
  max_iter = 1e+05,
  n_folds = 10,
  model_weights = c("Equal", "Proportional", "Stacking")[1]
)
```

Arguments

x	Design matrix.
y	Response vector.
n_models	Number of models into which the variables are split.
max_variables	Maximum number of variables that a model can contain.
keep	Proportion of models to keep based on their individual cross-validated errors. Default is 1.
model_criterion	Criterion for adding a variable to a model. Must be one of c("F-test", "RSS"). Default is "F-test".
stop_criterion	Criterion for determining when a model is saturated. Must be one of c("F-test", "pR2", "aR2", "R2", "Fixed"). Default is "F-test".
stop_parameter	Parameter value for the stopping criterion. Default is 0.05 for "F-test".
shrinkage	TRUE or FALSE parameter for shrinkage of the final models. Default is TRUE.
alpha	Elastic net mixing parmeter for model shrinkage. Default is 3/4.
include_intercept	TRUE or FALSE parameter for the inclusion of an intercept term.
n_lambda	Number of candidates for the sparsity penalty parameter. Default is 100.
tolerance	Convergence criteria for the coefficients. Default is 1e-3.
max_iter	Maximum number of iterations in the algorithm. Default is 1e5.

n_folds	Number of cross-validation folds. Default is 10.
model_weights	Criterion to determine the weights of the model for prediciton. Must be one of c("Equal", "Proportional", "Stacking"). Default is "Equal".

Value

An object of class stepSplitReg.

Author(s)

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

See Also

coef.stepSplitReg, predict.stepSplitReg

Examples

```
# Required Libraries
library(mvnfast)

# Setting the parameters
p <- 100
n <- 30
n.test <- 1000
sparsity <- 0.2
rho <- 0.5
SNR <- 3

# 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)))

# Correlation structure
Sigma <- matrix(0, p, p)
Sigma[1:p.active, 1:p.active] <- rho
diag(Sigma) <- 1
true.beta <- c(nonzero.betas, rep(0 , p - p.active))

# Computing the noise parameter for target SNR
sigma.epsilon <- as.numeric(sqrt((t(true.beta) %*% Sigma %*% true.beta)/SNR))

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

# Stepwise Split Regularized Regression
```

```
step.out <- stepSplitReg(x.train, y.train, n_models = 3, max_variables = NULL, keep = 4/4,
                         model_criterion = c("F-test", "RSS")[1],
                         stop_criterion = c("F-test", "pR2", "aR2", "R2", "Fixed")[1],
                         stop_parameter = 0.05,
                         shrinkage = TRUE, alpha = 4/4, include_intercept = TRUE,
                         n_lambda = 50, tolerance = 1e-2, max_iter = 1e5, n_folds = 5,
                         model_weights = c("Equal", "Proportional", "Stacking")[1])
step.coefficients <- coef(step.out, group_index = 1:step.out$n_models)
step.predictions <- predict(step.out, x.test, group_index = 1:step.out$n_models)
mspe.step <- mean((step.predictions-y.test)^2)/sigma.epsilon^2
```

Index

coef.cv.stepSplitReg, 2, *6*
coef.stepSplitReg, 3, *12*
cv.stepSplitReg, *2*, 5, *8*

predict.cv.stepSplitReg, *6*, 7
predict.stepSplitReg, 9, *12*

stepSplitReg, *4*, *9*, 10