Package ‘robStepSplitReg’

June 29, 2023

Type Package

Title Robust Stepwise Split Regularized Regression

Version 1.1.0

Date 2023-06-28

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

Description Functions to perform robust stepwise split regularized regression. The approach first uses a robust stepwise algorithm to split the variables into the models of an ensemble. An adaptive robust regularized estimator is then applied to each subset of predictors in the models of an ensemble.

License GPL (>= 2)

Encoding UTF-8

Biarch true

Imports Rcpp (>= 1.0.9), cellWise, glmnet

Suggests testthat, mvnfast

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

NeedsCompilation yes

Author Anthony Christidis [aut, cre],
Gabriela Cohen-Freue [aut]

Repository CRAN

Date/Publication 2023-06-29 21:40:02 UTC

R topics documented:

coef.robStepSplitReg *Coefficients for robStepSplitReg Object*

Description

coef.robStepSplitReg returns the coefficients for a robStepSplitReg object.

Usage

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

Arguments

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

Value

The coefficients for the robStepSplitReg object.

Author(s)

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

See Also

robStepSplitReg

Examples

```
# Required library
library(mvnfast)

# Simulation parameters
n <- 50
p <- 500
rho <- 0.5
p.active <- 100
snr <- 1
contamination.prop <- 0.2

# Setting the seed
set.seed(0)

# Simulation of beta vector
true.beta <- c(runif(p.active, 0, 5)*(-1)^rbinom(p.active, 1, 0.7), rep(0, p - p.active))
```

```
# Simulation of uncontaminated data
sigma.mat <- matrix(0, nrow = p, ncol = p)
sigma.mat[1:p.active, 1:p.active] <- rho
diag(sigma.mat) <- 1
x <- mvnfast::rmvn(n, mu = rep(0, p), sigma = sigma.mat)
sigma <- as.numeric(sqrt(t(true.beta) %*% sigma.mat %*% true.beta)/sqrt(snr))
y <- x %*% true.beta + rnorm(n, 0, sigma)

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) +
    k_lev * a / as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# Ensemble models
ensemble_fit <- robStepSplitReg(x_train, y_train,
                                n_models = 5,
                                model_saturation = c("fixed", "p-value")[1],
                                alpha = 0.05, model_size = n - 1,
                                robust = TRUE,
                                compute_coef = TRUE,
                                en_alpha = 1/4)

# Ensemble coefficients
ensemble_coefs <- coef(ensemble_fit, group_index = 1:ensemble_fit$n_models)
sens_ensemble <- sum(which((ensemble_coefs[-1]!=0)) <= p.active)/p.active
spec_ensemble <- sum(which((ensemble_coefs[-1]!=0)) <= p.active)/sum(ensemble_coefs[-1]!=0)

# Simulation of test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

# Prediction of test samples
ensemble_preds <- predict(ensemble_fit, newx = x_test,
                          group_index = 1:ensemble_fit$n_models,
                          dynamic = FALSE)
mspe_ensemble <- mean((y_test - ensemble_preds)^2)/sigma^2
```

predict.robStepSplitReg

Predictions for robStepSplitReg Object

Description

predict.robStepSplitReg returns the predictions for a robStepSplitReg object.

Usage

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

Arguments

object An object of class robStepSplitReg

newx New data for predictions.

group_index Groups included in the ensemble. Default setting includes all the groups.

dynamic Argument to determine whether dynamic predictions are used based on deviating cells. Default is FALSE.

... Additional arguments for compatibility.

Value

The predictions for the robStepSplitReg object.

Author(s)

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

See Also

robStepSplitReg

Examples

```
# Required library
library(mvnfast)

# Simulation parameters
n <- 50
p <- 500
rho <- 0.5
p.active <- 100
snr <- 1
contamination.prop <- 0.2
```

```
# Setting the seed
set.seed(0)

# Simulation of beta vector
true.beta <- c(runif(p.active, 0, 5)*(-1)^rbinom(p.active, 1, 0.7), rep(0, p - p.active))

# Simulation of uncontaminated data
sigma.mat <- matrix(0, nrow = p, ncol = p)
sigma.mat[1:p.active, 1:p.active] <- rho
diag(sigma.mat) <- 1
x <- mvnfast::rmvn(n, mu = rep(0, p), sigma = sigma.mat)
sigma <- as.numeric(sqrt(t(true.beta) %*% sigma.mat %*% true.beta)/sqrt(snr))
y <- x %*% true.beta + rnorm(n, 0, sigma)

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) +
    k_lev * a / as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# Ensemble models
ensemble_fit <- robStepSplitReg(x_train, y_train,
                                n_models = 5,
                                model_saturation = c("fixed", "p-value")[1],
                                alpha = 0.05, model_size = n - 1,
                                robust = TRUE,
                                compute_coef = TRUE,
                                en_alpha = 1/4)

# Ensemble coefficients
ensemble_coefs <- coef(ensemble_fit, group_index = 1:ensemble_fit$n_models)
sens_ensemble <- sum(which((ensemble_coefs[-1]!=0)) <= p.active)/p.active
spec_ensemble <- sum(which((ensemble_coefs[-1]!=0)) <= p.active)/sum(ensemble_coefs[-1]!=0)

# Simulation of test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

# Prediction of test samples
ensemble_preds <- predict(ensemble_fit, newx = x_test,
                          group_index = 1:ensemble_fit$n_models,
                          dynamic = FALSE)
mspe_ensemble <- mean((y_test - ensemble_preds)^2)/sigma^2
```

robStepSplitReg *Robust Stepwise Split Regularized Regression*

Description

robStepSplitReg performs robust stepwise split regularized regression.

Usage

```
robStepSplitReg(
  x,
  y,
  n_models = 1,
  model_saturation = c("fixed", "p-value")[1],
  alpha = 0.05,
  model_size = NULL,
  robust = TRUE,
  compute_coef = FALSE,
  en_alpha = 1/4
)
```

Arguments

x	Design matrix.
y	Response vector.
n_models	Number of models into which the variables are split.
model_saturation	Criterion to determine if a model is saturated. Must be one of "fixed" (default) or "p-value".
alpha	P-value used to determine when the model is saturated
model_size	Size of the models in the ensemble.
robust	Argument to determine if robust measures of location, scale and correlation are used. Default is TRUE.
compute_coef	Argument to determine if coefficients are computed (via adaptive PENSE) for each model. Default is FALSE.
en_alpha	Elastic net mixing parmeter for parameters shrinkage. Default is 1/4.

Value

An object of class robStepSplitReg.

Author(s)

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

See Also

coef.robStepSplitReg, predict.robStepSplitReg

Examples

```
# Required library
library(mvnfast)

# Simulation parameters
n <- 50
p <- 500
rho <- 0.5
p.active <- 100
snr <- 1
contamination.prop <- 0.2

# Setting the seed
set.seed(0)

# Simulation of beta vector
true.beta <- c(runif(p.active, 0, 5)*(-1)^rbinom(p.active, 1, 0.7), rep(0, p - p.active))

# Simulation of uncontaminated data
sigma.mat <- matrix(0, nrow = p, ncol = p)
sigma.mat[1:p.active, 1:p.active] <- rho
diag(sigma.mat) <- 1
x <- mvnfast::rmvn(n, mu = rep(0, p), sigma = sigma.mat)
sigma <- as.numeric(sqrt(t(true.beta) %*% sigma.mat %*% true.beta)/sqrt(snr))
y <- x %*% true.beta + rnorm(n, 0, sigma)

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) +
    k_lev * a / as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}
```

```
# Ensemble models
ensemble_fit <- robStepSplitReg(x_train, y_train,
                                n_models = 5,
                                model_saturation = c(”fixed”, ”p-value”)[1],
                                alpha = 0.05, model_size = n - 1,
                                robust = TRUE,
                                compute_coef = TRUE,
                                en_alpha = 1/4)

# Ensemble coefficients
ensemble_coefs <- coef(ensemble_fit, group_index = 1:ensemble_fit$n_models)
sens_ensemble <- sum(which((ensemble_coefs[-1]!=0)) <= p.active)/p.active
spec_ensemble <- sum(which((ensemble_coefs[-1]!=0)) <= p.active)/sum(ensemble_coefs[-1]!=0)

# Simulation of test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

# Prediction of test samples
ensemble_preds <- predict(ensemble_fit, newx = x_test,
                          group_index = 1:ensemble_fit$n_models,
                          dynamic = FALSE)
mspe_ensemble <- mean((y_test - ensemble_preds)^2)/sigma^2
```

Index