Package: SplitReg (via r-universe)

October 6, 2026

Type Package

Title Split Regularized Regression

Version 1.0.3

Date 2025-03-30

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

Description Functions for computing split regularized estimators defined in Christidis, Lakshmanan, Smucler and Zamar (2019) <doi:10.48550/arXiv.1712.03561>. The approach fits linear regression models that split the set of covariates into groups. The optimal split of the variables into groups and the regularized estimation of the regression coefficients are performed by minimizing an objective function that encourages sparsity within each group and diversity among them. The estimated coefficients are then pooled together to form the final fit.

License GPL (>= 2)

Biarch true

Imports Rcpp (>= 0.12.12)

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, glmnet, MASS

NeedsCompilation yes

RoxygenNote 7.3.2

Author Anthony Christidis [aut, cre], Ezequiel Smucler [aut], Ruben Zamar [aut]

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

Date/Publication 2025-03-30 18:40:01 UTC

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

RemoteRef HEAD

RemoteSha 8d0c742f1322e0f0b72acb726b03fd777ef04861

Contents

coef.cv.SplitReg 2
cv.SplitReg 3
predict.cv.SplitReg 5

Index **7**

`coef.cv.SplitReg` *Extract coefficients from a cv.SplitReg object.*

Description

Extract coefficients from a cv.SplitReg object.

Usage

```
## S3 method for class 'cv.SplitReg'
coef(object, index = object$index_opt, ...)
```

Arguments

`object`	Fitted cv.SplitReg object.
`index`	Indices indicating values of lambda_S at which to extract coefficients. Defaults to the optimal value.
`...`	Additional arguments for compatibility.

Value

A vector of coefficients

See Also

`cv.SplitReg`

Examples

```
library(MASS)
set.seed(1)
beta <- c(rep(5, 5), rep(0, 45))
Sigma <- matrix(0.5, 50, 50)
diag(Sigma) <- 1
x <- mvrnorm(50, mu = rep(0, 50), Sigma = Sigma)
y <- x %*% beta + rnorm(50)
fit <- cv.SplitReg(x, y, num_models=2)
split.coefs <- coef(fit)
```

`cv.SplitReg` *Split Regularized Regression algorithm with a sparsity and diversity penalty.*

Description

Computes a split regularized regression estimator. The sparsity and diversity penalty parameters are chosen automatically.

Usage

```
cv.SplitReg(
  x,
  y,
  num_lambdas_sparsity = 100,
  num_lambdas_diversity = 100,
  alpha = 1,
  num_models = 10,
  tolerance = 1e-08,
  max_iter = 1e+05,
  num_folds = 10,
  num_threads = 1
)
```

Arguments

`x`	Design matrix.
`y`	Response vector.
`num_lambdas_sparsity`	Length of the grid of sparsity penalties.
`num_lambdas_diversity`	Length of the grid of diversity penalties.
`alpha`	Elastic Net tuning constant: the value must be between 0 and 1. Default is 1 (Lasso).
`num_models`	Number of models to build.
`tolerance`	Tolerance parameter to stop the iterations while cycling over the models.
`max_iter`	Maximum number of iterations before stopping the iterations while cycling over the models.
`num_folds`	Number of folds for cross-validating.
`num_threads`	Number of threads used for parallel computation over the folds.

Details

Computes a split regularized regression estimator with `num_models` (G) models, defined as the linear models $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^G$ that minimize

$$\sum_{g=1}^{G} \left(\frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^g\|^2 + \lambda_S \left(\frac{(1-\alpha)}{2} \|\boldsymbol{\beta}^g\|_2^2 + \alpha \|\boldsymbol{\beta}\|_1 \right) + \frac{\lambda_D}{2} \sum_{h \neq g} \sum_{j=1}^{p} |\beta_j^h \beta_j^g| \right),$$

over grids for the penalty parameters λ_S and λ_D that are built automatically. Larger values of λ_S encourage more sparsity within the models and larger values of λ_D encourage more diversity among them. If $\lambda_D = 0$, then all of the models are equal to the Elastic Net regularized least squares estimator with penalty parameter λ_S. Optimal penalty parameters are found by `num_folds` cross-validation, where the prediction of the ensemble is formed by simple averaging. The predictors and the response are standardized to zero mean and unit variance before any computations are performed. The final output is in the original scales.

Value

An object of class cv.SplitReg, a list with entries

`betas`	Coefficients computed over the path of penalties for sparsity; the penalty for diversity is fixed at the optimal value.
`intercepts`	Intercepts for each of the models along the path of penalties for sparsity.
`index_opt`	Index of the optimal penalty parameter for sparsity.
`lambda_sparsity_opt`	Optimal penalty parameter for sparsity.
`lambda_diversity_opt`	Optimal penalty parameter for diversity.
`lambdas_sparsity`	Grid of sparsity parameters.
`lambdas_diversity`	Grid of diversity parameters.
`cv_mse_opt`	Optimal CV MSE.
`call`	The matched call.

See Also

`predict.cv.SplitReg`, `coef.cv.SplitReg`

Examples

```
library(MASS)
set.seed(1)
beta <- c(rep(5, 5), rep(0, 45))
Sigma <- matrix(0.5, 50, 50)
diag(Sigma) <- 1
x <- mvrnorm(50, mu = rep(0, 50), Sigma = Sigma)
y <- x %*% beta + rnorm(50)
```

```
fit <- cv.SplitReg(x, y, num_models=2)
coefs <- predict(fit, type="coefficients")
```

`predict.cv.SplitReg` *Make predictions from a cv.SplitReg object.*

Description

Make predictions from a cv.SplitReg object, similar to other predict methods.

Usage

```
## S3 method for class 'cv.SplitReg'
predict(
  object,
  newx,
  index = object$index_opt,
  type = c("response", "coefficients"),
  ...
)
```

Arguments

`object`	Fitted cv.SplitReg object.
`newx`	Matrix of new values of x at which prediction are to be made. Ignored if type is "coefficients".
`index`	Indices indicating values of lambda_S at which to predict. Defaults to the optimal value.
`type`	Either "response" for predicted values or "coefficients" for the estimated coefficients.
`...`	Additional arguments for compatibility.

Value

Either a matrix with predictions or a vector of coefficients

See Also

`predict.cv.SplitReg`

Examples

```
library(MASS)
set.seed(1)
beta <- c(rep(5, 5), rep(0, 45))
Sigma <- matrix(0.5, 50, 50)
diag(Sigma) <- 1
x <- mvrnorm(50, mu = rep(0, 50), Sigma = Sigma)
y <- x %*% beta + rnorm(50)
fit <- cv.SplitReg(x, y, num_models=2)
x.new <- mvrnorm(50, mu = rep(0, 50), Sigma = Sigma)
split.predictions <- predict(fit, newx = x.new, type="response")
```

Index

coef.cv.SplitReg, 2, *4*
cv.SplitReg, *2*, 3

predict.cv.SplitReg, *4*, 5, *5*