Package 'QuantRegGLasso'

January 16, 2024

Title Adaptively Weighted Group Lasso for Semiparametric Quantile Regression Models

Version 1.0.0

Description Implements an adaptively weighted group Lasso procedure for simultaneous variable selection and structure identification in varying coefficient quantile regression models and additive quantile regression models with ultra-high dimensional covariates. The methodology, grounded in a strong sparsity condition, establishes selection consistency under certain weight conditions. To address the challenge of tuning parameter selection in practice, a BIC-type criterion named high-dimensional information criterion (HDIC) is proposed. The Lasso procedure, guided by HDIC-determined tuning parameters, maintains selection consistency. Theoretical findings are strongly supported by simulation studies. (Toshio Honda, Ching-Kang Ing, Wei-Ying Wu, 2019, <DOI:10.3150/18-BEJ1091>).

License GPL (>= 2)

ByteCompile true

BugReports https://github.com/egpivo/QuantRegGLasso/issues

Depends R (>= 3.4.0)

Imports Rcpp (>= 1.0.12), ggplot2

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, testthat (>= 2.1.0)

SystemRequirements GNU make

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/egpivo/SpatPCA

Config/testthat/edition 3

NeedsCompilation yes

Author Wen-Ting Wang [aut, cre] (<https://orcid.org/0000-0003-3051-7302>),
Wei-Ying Wu [aut],
Toshio Honda [aut],
Ching-Kang Ing [aut] (<https://orcid.org/0000-0003-1362-8246>)

Maintainer Wen-Ting Wang <egpivo@gmail.com>
Repository CRAN
Date/Publication 2024-01-16 17:20:02 UTC

R topics documented:

orthogonize_bspline 2
plot.qrglasso 3
plot.qrglasso.predict 4
predict 5
qrglasso 6

Index **9**

`orthogonize_bspline` *Orthogonalized B-splines*

Description

Generate a set of orthogonalized B-splines using the Gram-Schmidt algorithm applied to the built-in function `splines::bs()`.

Usage

```
orthogonize_bspline(
  knots,
  boundary_knots,
  degree,
  predictors = NULL,
  is_approx = FALSE
)
```

Arguments

`knots`	Array. The knots that define the spline.
`boundary_knots`	Array. The breakpoints that define the spline.
`degree`	Integer. The degree of the piecewise polynomial.
`predictors`	Array. The predictor variables with size p.
`is_approx`	Boolean. The default is `FALSE`.

Value

A list containing:

`bsplines`	Matrix of orthogonalized B-splines with dimensions $(p, \text{length}(knots)+\text{degree}+1)$
`z`	Predictors used in generation

Examples

```
# Example: Generate and plot the first 5 orthogonalized B-splines
p <- 30
total_knots <- 10
degree <- 3
boundaries <- c(0, 1)
x <- seq(from = 0, to = 1, length.out = total_knots)
knots <- x[2:(total_knots - 1)]
predictors <- runif(p, min = 0, max = 1)
bsplines <- orthogonize_bspline(knots, boundaries, degree, predictors)

# Plot the first 5 B-splines
index <- order(bsplines$z)
original_par <- par(no.readonly = TRUE)
par(mfrow = c(1, 5))
for (i in 1:5)
  plot(bsplines$z[index], bsplines$bsplines[index, i], main = i, type = "l")
par(original_par)
```

---

plot.qrglasso *Display BIC Results from* qrglasso

Description

Visualize the HDIC BIC results corresponding to hyperparameters obtained from qrglasso.

Usage

```
## S3 method for class 'qrglasso'
plot(x, ...)
```

Arguments

x	An object of class qrglasso for the plot method.
...	Additional parameters not used directly.

Value

NULL

See Also

qrglasso

Examples

```
set.seed(123)
n <- 100
p <- 5
L <- 5
Y <- matrix(rnorm(n), n, 1)
W <- matrix(rnorm(n * p * (L - 1)), n, p * (L - 1))

# Call qrglasso with default parameters
result <- qrglasso(Y = Y, W = W, p = p)

# Visualize the BIC results
plot(result)
```

---

plot.qrglasso.predict *Display Predicted Coefficient Functions from* qrglasso

---

Description

Visualize the predicted coefficient functions selected by BIC.

Usage

```
## S3 method for class 'qrglasso.predict'
plot(x, ...)
```

Arguments

x	An object of class qrglasso.predict for the plot method.
...	Additional parameters not used directly.

Value

NULL

See Also

qrglasso

Examples

```
set.seed(123)
n <- 100
p <- 5
L <- 5
Y <- matrix(rnorm(n), n, 1)
W <- matrix(rnorm(n * p * (L - 1)), n, p * (L - 1))
```

```
# Call qrglasso with default parameters
result <- qrglasso(Y = Y, W = W, p = p)

# Predict the top-k coefficient functions
estimate <- predict(result, top_k = 2)

# Display the predicted coefficient functions
plot(estimate)
```

`predict` *Predict Top-k Coefficient Functions*

Description

Predict the top-k coefficient functions based on a `qrglasso` class object.

Usage

```
predict(
  qrglasso_object,
  metric_type = "BIC",
  top_k = 5,
  degree = 2,
  boundaries = c(0, 1),
  is_approx = FALSE
)
```

Arguments

`qrglasso_object`	A `qrglasso` class object.
`metric_type`	Character. Metric type for gamma selection, e.g., `BIC`, `BIC-log`. Default is `BIC`.
`top_k`	Integer. The number of top estimated functions to predict. Default is 5.
`degree`	Integer. Degree of the piecewise polynomial. Default is 2.
`boundaries`	Array. Two boundary points for the piecewise polynomial. Default is c(0, 1).
`is_approx`	Logical. If TRUE, the size of covariate indexes will be 1e6; otherwise, 1e4. Default is FALSE.

Value

A list containing:

`coef_functions`	Matrix. The estimated top-k coefficient functions with dimension $(m \times k)$, where m is the size of `z`.
`z`	Array. Index predictors used in the generation.

See Also

qrglasso

Examples

```
set.seed(123)
n <- 100
p <- 5
L <- 5
Y <- matrix(rnorm(n), n, 1)
W <- matrix(rnorm(n * p * (L - 1)), n, p * (L - 1))

# Call qrglasso with default parameters
result <- qrglasso(Y = Y, W = W, p = p)
estimate <- predict(result)
print(dim(estimate$coef_functions))
```

---

qrglasso *Adaptively Weighted Group Lasso*

Description

The function qrglasso performs Adaptively Weighted Group Lasso for semiparametric quantile regression models. It estimates the coefficients of a quantile regression model with adaptively weighted group lasso regularization. The algorithm supports the use of B-spline basis functions to model the relationship between covariates and the response variable. Regularization is applied across different groups of covariates, and an adaptive weighting scheme is employed to enhance variable selection.

Usage

```
qrglasso(
  Y,
  W,
  p,
  omega = NULL,
  tau = 0.5,
  qn = 1,
  lambda = NULL,
  maxit = 1000,
  thr = 1e-04
)
```

Arguments

Y	A $n \times 1$ data matrix where n is the sample size.
W	A $n \times (p \times L)$ B-spline matrix where L is the number of groups and p is the number of covariates.
p	A numeric indicating the number of covariates.
omega	A $p \times 1$ weight matrix. Default value is NULL.
tau	A numeric quantile of interest. Default value is 0.5.
qn	A numeric bound parameter for HDIC. Default value is 1.
lambda	A sequence of tuning parameters. Default value is NULL.
maxit	The maximum number of iterations. Default value is 1000.
thr	Threshold for convergence. Default value is 10^{-4}.

Value

A list with the following components:

gamma	A target estimate.
xi	An auxiliary estimate in the ADMM algorithm.
phi	An auxiliary estimate in the ADMM algorithm.
BIC	A sequence of BIC values with respect to different lambdas.
lambda	A sequence of tuning parameters used in the algorithm.
L	The number of groups.
omega	A $p \times 1$ weight matrix used in the algorithm.

Author(s)

Wen-Ting Wang

References

Toshio Honda, Ching-Kang Ing, Wei-Ying Wu (2019). Adaptively weighted group Lasso for semi-parametric quantile regression models. *Bernoulli* **225** 4B.

Examples

```
# Example: One true non-linear covariate function
# Define the function g1
g1 <- function(x) {
  (3 * sin(2 * pi * x) / (2 - sin(2 * pi * x))) - 0.4641016
}

# Set parameters
n <- 100
p <- 50
err_sd <- 0.1 ** 2
tau <- 0.7
```

```
# Generate synthetic data
set.seed(1234)
x <- matrix(runif(n * p, min = 0, max = 1), n, p)
error_tau <- rnorm(n, sd = err_sd) - qnorm(tau, sd = err_sd)
y <- g1(x[, 1]) + error_tau
y <- y - mean(y)

# B-spline parameters
total_knots <- 5
degree <- 2
boundaries <- c(0, 1)
xx <- seq(from = 0, to = 1, length.out = total_knots)
knots <- xx[2:(total_knots - 1)]

# Create B-spline matrix W
L <- total_knots + degree - 1
bspline_results <- lapply(1:n, function(i) orthogonize_bspline(knots, boundaries, degree, x[i, ]))
W <- matrix(
   t(sapply(bspline_results, function(result) sqrt(L) * result$bsplines[, -1])),
   ncol = p * (L - 1),
   byrow = TRUE
)

# Perform quantile regression with group Lasso
n_lambda <- 10
max_lambda <- 10
lambda <- c(0, exp(seq(log(max_lambda / 1e4), log(max_lambda), length = (n_lambda - 1))))
result <- qrglasso(as.matrix(y), W, p)
# BIC Results
plot(result)
# Prediction
estimate = predict(result, top_k = 1)
plot(estimate)
```

Index

orthogonize_bspline, 2

plot.qrglasso, 3
plot.qrglasso.predict, 4
predict, 5

qrglasso, *3*, *4*, *6*, 6