Package 'penalizedcdf'

January 30, 2023

Type Package

Title Estimate a Penalized Linear Model using the CDF Penalty Function

Version 0.1.0

Author Daniele Cuntrera [aut, cre],
Luigi Augugliaro [aut],
Vito M.R. Muggeo [aut]

Maintainer Daniele Cuntrera <daniele.cuntrera@unipa.it>

Description Utilize the CDF penalty function to estimate a penalized linear model.
It enables you to display some graphical representations and determine whether the Karush-Kuhn-Tucker conditions are met.
For more details about the theory, please refer to Cuntrera, D., Augugliaro, L., & Muggeo, V. M. (2022) <arXiv:2212.08582>.

License GPL-2 | GPL-3

Encoding UTF-8

Imports plot.matrix

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-01-30 16:40:02 UTC

R topics documented:

BIC_calc *BIC calculator function*

Description

Function that takes the resulting values of the estimated model as input, to compute BIC

Usage

```
BIC_calc(X,
         b.tld,
         y,
         n)
```

Arguments

X	The covariates' matrix
b.tld	The estimated sparse-beta
y	The response variable
n	The number of observation

Value

Returns the BIC value calculated for a single value of the tuning parameter.

Examples

```
p <- 10
n <- 100
X <- cbind(1, matrix(rnorm(n * p), n , p))
b.s <- c(1, rep(0, p))
b.s[sample(2:p, 3)] <- 1
y <- drop(crossprod(t(X), b.s))
out <- cdfPen(X = X, y = y)


(bic <- BIC_cdfpen(out))
plot(out$lmb, bic, "s")
```

BIC_cdfpen — *BIC computation from a "cdfpen" object*

Description

Calculates the BIC for all estimated models in a "cdfpen" object

Usage

```
BIC_cdfpen(object)
```

Arguments

`object`	Object containing the results.

Value

Returns a vector containing the BIC values calculated over the entire estimated path

cdfPen — *Fit a Linear Model with with CDF regularization*

Description

Uses the CDF penalty to estimate a linear model with the maximum penalized likelihood. The path of coefficients is computed for a grid of values for the lambda regularization parameter.

Usage

```
cdfPen(X,
       y,
       nu,
       lmb,
       nlmb = 100L,
       e = 1E-3,
       rho = 2,
       algorithm = c("lla", "opt"),
       nstep = 1E+5,
       eps = 1E-6,
       eps.lla = 1E-6,
       nstep.lla = 1E+5)
```

Arguments

`X`	Matrix of covariates, each row is a vector of observations. The matrix must not contain the intercept.
`y`	Vector of response variable.
`nu`	Shape parameter of the penalty. It affects the degree of the non-convexity of the penalty. If no value is specified, the smallest value that ensures a single solution will be used.
`lmb`	A user-supplied tuning parameter sequence.
`nlmb`	number of lambda values; 100 is the default value.
`e`	The smallest lambda value, expressed as a percentage of maximum lambda. Default value is .001.
`rho`	Parameter of the optimization algorithm. Default is 2.
`algorithm`	Approximation to be used to obtain the sparse solution.
`nstep`	Maximum number of iterations of the global algorithm.
`eps`	Convergence threshold of the global algorithm.
`eps.lla`	Convergence threshold of the LLA-algorithm (if used).
`nstep.lla`	Maximum number of iterations of the LLA-algorithm (if used).

Details

We consider a local quadratic approximation of the likelihood to treat the problem as a weighted linear model.

The choice of value assigned to ν is of fundamental importance: it affects both computational and estimation aspects. It affects the "degree of non-convexity" of the penalty and determines which of the good and bad properties of convex and non-convex penalties are obtained. Using a high value of ν ensures the uniqueness of solution, but the estimates will be biased. Conversely, a small value of ν guarantees negligible bias in the estimates. The parameter ν has the role of determining the convergence rate of non-null estimates$: the lower the value, the higher the convergence rate. Using lower values of ν, the objective function will have local minima.

Value

`coefficients`	The coefficients fit matrix. The number of columns is equal to nlmb, and the number of rows is equal to the number of coefficients.
`lmb`	The vector of lambda used.
`e`	The smallest lambda value, expressed as a percentage of maximum lambda. Default value is .001.
`rho`	The parameter of the optimization algorithm used
`nu`	The shape parameters of the penalty used.
`X`	The design matrix.
`y`	The response.
`algorithm`	Approximation used

Author(s)

Daniele Cuntrera, Luigi Augugliaro, Vito Muggeo

References

Aggiungere Arxiv

Examples

```
p <- 10
n <- 100
X <- cbind(1, matrix(rnorm(n * p), n , p))
b.s <- c(1, rep(0, p))
b.s[sample(2:p, 3)] <- 1
y <- drop(crossprod(t(X), b.s))
out <- cdfPen(X = X, y = y)
```

`cdfPen.fit` *Fitter function for CDF penalty*

Description

These are the fundamental computing algorithms that cdfPen invokes to estimate penalized linear models by varying lambda.

Usage

```
cdfPen.fit(b,
           b.tld,
           g,
           b.rho,
           H.rho,
           lmb.rho,
           nu,
           algorithm,
           nstep = 1E+5,
           eps = 1E-5,
           eps.lla = 1E-6,
           nstep.lla = 1E+5)
```

Arguments

`b`	Starting values of beta-vector.
`b.tld`	Starting values of sparse beta-vector.
`g`	Starting values of pseudo-variable.
`b.rho`	Ridge solution.

`H.rho`	Second part of ridge solution.
`lmb.rho`	Lambda-rho ratio.
`nu`	Shape parameter of the penalty. It affects the degree of the non-convexity of the penalty.
`algorithm`	Approximation to be used to obtain the sparse solution.
`nstep`	Maximum number of iterations of the global algorithm.
`eps`	Convergence threshold of the global algorithm.
`eps.lla`	Convergence threshold of the LLA-algorithm (if used).
`nstep.lla`	Maximum number of iterations of the LLA-algorithm (if used).

Value

`b`	Estimated beta-vector.
`b.tld`	Estimated sparse beta-vector.
`g`	Final values of pseudo-variable.
`i`	Number of iterations.
`conv`	Convergence check status (0 if converged).

Author(s)

Daniele Cuntrera, Luigi Augugliaro, Vito Muggeo

References

Aggiungere Arxiv

`check_KKT` *Check on the condition of Karush-Kuhn-Tucker*

Description

Control over Karush-Kuhn-Tucker (Karush, 1939) conditions for the estimates obtained.

Usage

```
check_KKT(obj,
          intercept = TRUE)
```

Arguments

`obj`	Object to be checked.
`intercept`	Is the intercept used in the model?

Value

grd	The value of gradient.
hx	The value of equality constraint.
glob	The global value of derivative (grd + hx).
test	Is the condition verified?
lmb	The values of lambda used in the model

Author(s)

Daniele Cuntrera, Luigi Augugliaro, Vito Muggeo

References

Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago.

Examples

```
p <- 10
n <- 100
X <- cbind(1, matrix(rnorm(n * p), n , p))
b.s <- c(1, rep(0, p))
b.s[sample(2:p, 3)] <- 1
y <- drop(crossprod(t(X), b.s))
out <- cdfPen(X = X, y = y)

KKT <- check_KKT(out)
plot(KKT$test)
```

---

lla — *LLA approximation for CDF penalty*

Description

Linearly approximate a part of the objective function to greatly speed up computations.

Usage

```
lla(b.o,
    lmb.rho,
    bm_gm,
    nu,
    nstep.lla = 100L,
    eps.lla = 1E-6)
```

Arguments

`b.o`	Vector of sparse-solution.
`lmb.rho`	Lambda-rho ratio.
`bm_gm`	Vector of pseudo-solution
`nu`	Shape parameter of the penalty.
`nstep.lla`	Maximum number of iterations of the LLA-algorithm (if used).
`eps.lla`	Convergence threshhold of the LLA-algorithm (if used).

Details

The LLA approximation allows the computationally intensive part to be treated as a weighted LASSO (Tibshirani, 1996) problem. In this way the computational effort is significantly less while maintaining satisfactory accuracy of the results. See Zou and Li (2008).

Value

`b`	Vector of the estimated sparse-solution.
`Conv`	Convergence check (0 if converged).
`nstep.lla`	Number of iterations done.

References

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288.

Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. Annals of statistics, 36(4):1509

`plot_cdfpen` *Plot coefficients or BIC from a "cdfpen" object*

Description

Plot coefficient profile plot or BIC trend

Usage

```
plot_cdfpen(object,
            ...)
```

Arguments

`object`	Object to be plotted.
`...`	Other graphical parameters to plot.

Details

A graph showing the BIC trend or profile of coefficients is displayed.

Value

No return value

Author(s)

Daniele Cuntrera, Luigi Augugliaro, Vito Muggeo

Examples

```
p <- 10
n <- 100
X <- cbind(1, matrix(rnorm(n * p), n , p))
b.s <- c(1, rep(0, p))
b.s[sample(2:p, 3)] <- 1
y <- drop(crossprod(t(X), b.s))
out <- cdfPen(X = X, y = y)

plot_cdfpen(out)         #Coefficients' path ~ lambda
plot_cdfpen(out, "l1")   #Coefficients' path ~ L1 norm
plot_cdfpen(out, "BIC")  #BIC ~ lambda
```

---

plot_path — *Plotter function for cdfpen class*

Description

Function that takes user requests as input, to show the requested graph

Usage

```
plot_path(obj,
          lmb,
          coeff,
          type = c("path", "l1", "BIC"),
          ...)
```

Arguments

obj	Object to be plotted
lmb	lambda values used in the model
coeff	the coefficients' matrix
type	type of graph to be ploted
...	Other characteristics to be added

Value

No return value

S *Threshold function for CDF penalty*

Description

Applies the threshold rule to obtain the vector of sparse estimates

Usage

```
S(bm_gm,
  db,
  w)
```

Arguments

`bm_gm`	Vector of pseudo-solution.
`db`	Lambda-rho ratio.
`w`	Weights obtained from the penalty function.

Value

The estimated coefficient

Index