Package: simest (via r-universe)

March 24, 2025

Title Constrained Single Index Model Estimation

Type Package

LazyLoad yes

LazyData yes

Version 0.4

Author Arun Kumar Kuchibhotla <arunku@wharton.upenn.edu>, Rohit Kumar Patra <rohit@stat.columbia.edu>

Maintainer Arun Kumar Kuchibhotla <arunku@wharton.upenn.edu>

Date 2017-04-08.

Depends nnls, cobs

Description Estimation of function and index vector in single index model with and without shape constraints including different smoothness conditions.

License GPL-2

NeedsCompilation yes

Date/Publication 2017-04-25 14:35:06 UTC

Additional_repositories https://cranhaven.r-universe.dev

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

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/simest

RemoteSha 08892e7bddb4a7777b76dd29c8b9bd6c3651cbbd

RemoteSubdir simest

Contents

---

cpen *C code for convex penalized least squares regression.*

Description

This function is only intended for an internal use.

Usage

```
cpen(dim, t_input, z_input, w_input, a0_input,
 lambda_input, Ky_input, L_input, U_input,
 fun_input, res_input, flag, tol_input,
 zhat_input, iter, Deriv_input)
```

Arguments

`dim`	vector of sample size and maximum iteration.
`t_input`	x-vector in cvx.pen.reg.
`z_input`	y-vector in cvx.pen.reg.
`w_input`	w-vector in cvx.pen.reg.
`a0_input`	initial vector for iterative algorithm.
`lambda_input`	lambda-value in cvx.pen.reg.
`Ky_input`	Internal vector used for algorithm.
`L_input`	Internal vector. Set to 0.
`U_input`	Internal vector. Set to 0.
`fun_input`	Internal vector. Set to 0.
`res_input`	Internal vector. Set to 0.
`flag`	Logical for stop criterion.
`tol_input`	tolerance level used in cvx.pen.reg.
`zhat_input`	Internal vector. Set to zero. Stores the final output.
`iter`	Iteration number inside the algorithm.
`Deriv_input`	Internal vector. Set to zero. Stores the derivative vector.

Details

See the source for more details about the algorithm.

Value

Does not return anything. Changes the inputs according to the iterations.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

Source

Dontchev, A. L., Qi, H. and Qi, L. (2003). Quadratic Convergence of Newton's Method for Convex Interpolation and Smoothing. Constructive Approximation, 19(1):123-143.

---

cvx.lip.reg *Convex Least Squares Regression with Lipschitz Constraint*

Description

This function provides an estimate of the non-parametric regression function with a shape constraint of convexity and a smoothness constraint via a Lipschitz bound.

Usage

```
cvx.lip.reg(t, z, w = NULL, L,...)
## Default S3 method:
cvx.lip.reg(t, z, w = NULL, L, ...)
## S3 method for class 'cvx.lip.reg'
plot(x,...)
## S3 method for class 'cvx.lip.reg'
print(x,...)
## S3 method for class 'cvx.lip.reg'
predict(object, newdata = NULL, deriv = 0, ...)
```

Arguments

t	a numeric vector giving the values of the predictor variable.
z	a numeric vector giving the values of the response variable.
w	an optional numeric vector of the same length as x; Defaults to all elements $1/n$.
L	a numeric value providing the Lipschitz bound on the function.
...	additional arguments.
x	an object of class 'cvx.lip.reg'.
object	An object of class 'cvx.lip.reg'.
newdata	a matrix of new data points in the predict function.
deriv	a numeric either 0 or 1 representing which derivative to evaluate.

Details

The function minimizes

$$\sum_{i=1}^{n} w_i(z_i - \theta_i)^2$$

subject to

$$-L \leq \frac{\theta_2 - \theta_1}{t_2 - t_1} \leq \cdots \leq \frac{\theta_n - \theta_{n-1}}{t_n - t_{n-1}} \leq L$$

for sorted t values and z reorganized such that z_i corresponds to the new sorted t_i. This function uses the `nnls` function from the `nnls` package to perform the constrained minimization of least squares. `plot` function provides the scatterplot along with fitted curve; it also includes some diagnostic plots for residuals. Predict function now allows calculating the first derivative also.

Value

An object of class 'cvx.lip.reg', basically a list including the elements

`x.values`	sorted 't' values provided as input.
`y.values`	corresponding 'z' values in input.
`fit.values`	corresponding fit values of same length as that of 'x.values'.
`deriv`	corresponding values of the derivative of same length as that of 'x.values'.
`residuals`	residuals obtained from the fit.
`minvalue`	minimum value of the objective function attained.
`iter`	Always set to 1.
`convergence`	a numeric indicating the convergence of the code.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

Source

Lawson, C. L and Hanson, R. J. (1995). Solving Least Squares Problems. SIAM.

References

Chen, D. and Plemmons, R. J. (2009). Non-negativity Constraints in Numerical Analysis. Symposium on the Birth of Numerical Analysis.

See Also

See also the function `nnls`.

Examples

```
args(cvx.lip.reg)
x <- runif(50,-1,1)
y <- x^2 + rnorm(50,0,0.3)
tmp <- cvx.lip.reg(x, y, L = 10)
print(tmp)
plot(tmp)
predict(tmp, newdata = rnorm(10,0,0.1))
```

`cvx.lse.con.reg` *Convex Least Squares Regression.*

Description

This function provides an estimate of the non-parametric regression function with a shape constraint of convexity and no smoothness constraint. Note that convexity by itself provides some implicit smoothness.

Usage

```
cvx.lse.con.reg(t, z, w = NULL,...)
## Default S3 method:
cvx.lse.con.reg(t, z, w = NULL, ...)
```

Arguments

`t`	a numeric vector giving the values of the predictor variable.
`z`	a numeric vector giving the values of the response variable.
`w`	an optional numeric vector of the same length as t; Defaults to all elements $1/n$.
`...`	additional arguments.

Details

This function does the same thing as `cvx.lse.reg` except that here we use conreg function from `cobs` package which is faster than `cvx.lse.reg`. The plot, predict, print functions of cvx.lse.reg also apply for cvx.lse.con.reg.

Value

An object of class 'cvx.lse.reg', basically a list including the elements

`x.values`	sorted 't' values provided as input.
`y.values`	corresponding 'z' values in input.
`fit.values`	corresponding fit values of same length as that of 'x.values'.
`deriv`	corresponding values of the derivative of same length as that of 'x.values'.
`iter`	number of steps taken to complete the iterations.

residuals	residuals obtained from the fit.
minvalue	minimum value of the objective function attained.
convergence	a numeric indicating the convergence of the code. Always set to 1.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu

Source

Lawson, C. L and Hanson, R. J. (1995). Solving Least Squares Problems. SIAM.

References

Chen, D. and Plemmons, R. J. (2009). Non-negativity Constraints in Numerical Analysis. Symposium on the Birth of Numerical Analysis.

Liao, X. and Meyer, M. C. (2014). coneproj: An R package for the primal or dual cone projections with routines for constrained regression. Journal of Statistical Software 61(12), 1 – 22.

Examples

```
args(cvx.lse.con.reg)
x <- runif(50,-1,1)
y <- x^2 + rnorm(50,0,0.3)
tmp <- cvx.lse.con.reg(x, y)
print(tmp)
plot(tmp)
predict(tmp, newdata = rnorm(10,0,0.1))
```

---

cvx.lse.reg *Convex Least Squares Regression.*

Description

This function provides an estimate of the non-parametric regression function with a shape constraint of convexity and no smoothness constraint. Note that convexity by itself provides some implicit smoothness.

Usage

```
cvx.lse.reg(t, z, w = NULL,...)
## Default S3 method:
cvx.lse.reg(t, z, w = NULL, ...)
## S3 method for class 'cvx.lse.reg'
plot(x,...)
## S3 method for class 'cvx.lse.reg'
print(x,...)
## S3 method for class 'cvx.lse.reg'
predict(object, newdata = NULL, deriv = 0, ...)
```

Arguments

t	a numeric vector giving the values of the predictor variable.
z	a numeric vector giving the values of the response variable.
w	an optional numeric vector of the same length as t; Defaults to all elements $1/n$.
...	additional arguments.
x	An object of class 'cvx.lse.reg'. This is for plot and print function.
object	An object of class 'cvx.lse.reg'.
newdata	a matrix of new data points in the predict function.
deriv	a numeric either 0 or 1 representing which derivative to evaluate.

Details

The function minimizes

$$\sum_{i=1}^{n} w_i (z_i - \theta_i)^2$$

subject to

$$\frac{\theta_2 - \theta_1}{t_2 - t_1} \leq \cdots \leq \frac{\theta_n - \theta_{n-1}}{t_n - t_{n-1}}$$

for sorted t values and z reorganized such that z_i corresponds to the new sorted t_i. This function previously used the `coneA` function from the `coneproj` package to perform the constrained minimization of least squares. Currently, the code makes use of the `nnls` function from `nnls` package for the same purpose. `plot` function provides the scatterplot along with fitted curve; it also includes some diagnostic plots for residuals. Predict function now allows computation of the first derivative.

Value

An object of class 'cvx.lse.reg', basically a list including the elements

x.values	sorted 't' values provided as input.
y.values	corresponding 'z' values in input.
fit.values	corresponding fit values of same length as that of 'x.values'.
deriv	corresponding values of the derivative of same length as that of 'x.values'.
iter	number of steps taken to complete the iterations.
residuals	residuals obtained from the fit.
minvalue	minimum value of the objective function attained.
convergence	a numeric indicating the convergence of the code.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu

Source

Lawson, C. L and Hanson, R. J. (1995). Solving Least Squares Problems. SIAM.

References

Chen, D. and Plemmons, R. J. (2009). Non-negativity Constraints in Numerical Analysis. Symposium on the Birth of Numerical Analysis.

Liao, X. and Meyer, M. C. (2014). coneproj: An R package for the primal or dual cone projections with routines for constrained regression. Journal of Statistical Software 61(12), 1 – 22.

Examples

```
args(cvx.lse.reg)
x <- runif(50,-1,1)
y <- x^2 + rnorm(50,0,0.3)
tmp <- cvx.lse.reg(x, y)
print(tmp)
plot(tmp)
predict(tmp, newdata = rnorm(10,0,0.1))
```

---

cvx.pen.reg *Penalized Smooth Convex Regression.*

Description

This function provides an estimate of the non-parametric regression function with a shape constraint of convexity and smoothness constraint provided through square integral of second derivative.

Usage

```
cvx.pen.reg(x, y, lambda, w = NULL, tol = 1e-05, maxit = 1000,...)
## Default S3 method:
cvx.pen.reg(x, y, lambda, w = NULL, tol = 1e-05, maxit = 1000,...)
## S3 method for class 'cvx.pen.reg'
plot(x,...)
## S3 method for class 'cvx.pen.reg'
print(x,...)
## S3 method for class 'cvx.pen.reg'
predict(object, newdata = NULL,...)
```

Arguments

x	a numeric vector giving the values of the predictor variable. For plot and print functions, x represents an object of class cvx.pen.reg.
y	a numeric vector giving the values of the response variable.
lambda	a numeric value giving the penalty value.
w	an optional numeric vector of the same length as x; Defaults to all 1.
maxit	an integer giving the maxmimum number of steps taken by the algorithm; defaults to 1000.

tol	a numeric providing the tolerance level for convergence.
...	any additional arguments.
object	An object of class 'cvx.pen.reg'. This is for predict function.
newdata	a vector of new data points to be used in the predict function.

Details

The function minimizes

$$\sum_{i=1}^{n} w_i(y_i - f(x_i))^2 + \lambda \int \{f''(x)\}^2 dx$$

subject to convexity constraint on f. plot function provides the scatterplot along with fitted curve; it also includes some diagnostic plots for residuals. Predict function returns a matrix containing the inputted newdata along with the function values, derivatives and second derivatives.

Value

An object of class 'cvx.pen.reg', basically a list including the elements

x.values	sorted 'x' values provided as input.
y.values	corresponding 'y' values in input.
fit.values	corresponding fit values of same length as that of 'x.values'.
deriv	corresponding values of the derivative of same length as that of 'x.values'.
iter	number of steps taken to complete the iterations.
residuals	residuals obtained from the fit.
minvalue	minimum value of the objective function attained.
convergence	a numeric indicating the convergence of the code.
alpha	a numeric vector of length 2 less than 'x'. This represents the coefficients of the B-splines in the second derivative of the estimator.
AlphaMVal	a numeric vector needed for predict function.
lower	a numeric vector needed for predict function.
upper	a numeric vector needed for predict function.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu, Rohit Kumar Patra, rohit@stat.columbia.edu.

Source

Elfving, T. and Andersson, L. (1988). An Algorithm for Computing Constrained Smoothing Spline Functions. Numer. Math., 52(5):583–595.

Dontchev, A. L., Qi, H. and Qi, L. (2003). Quadratic Convergence of Newton's Method for Convex Interpolation and Smoothing. Constructive Approximation, 19(1):123-143.

Examples

```
args(cvx.pen.reg)
x <- runif(50,-1,1)
y <- x^2 + rnorm(50,0,0.3)
tmp <- cvx.pen.reg(x, y, lambda = 0.01)
print(tmp)
plot(tmp)
predict(tmp, newdata = rnorm(10,0,0.1))
```

`derivcvxpec` *C code for prediction using cvx.lse.reg, cvx.lip.reg and cvx.lse.con.reg.*

Description

This function is only intended for an internal use.

Usage

```
derivcvxpec(dim, t, zhat, D, kk)
```

Arguments

`dim`	vector of sample size, size of newdata and which derivative to compute.
`t`	x-vector in cvx.lse.reg and others.
`zhat`	prediction obtained from cvx.lse.reg and others.
`D`	derivative vector obtained from cvx.lse.reg and others.
`kk`	vector storing the final prediction.

Details

The estimate is a linear interpolator and the algorithm implements this.

Value

Does not return anything. Changes the inputs according to the algorithm.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

`fastmerge` *Pre-binning of Data Points.*

Description

Numerical tolerance problems in non-parametric regression makes it necessary for pre-binning of data points. This procedure is implicitly performed by most of the regression function in R. This function implements this procedure with a given tolerance level.

Usage

```
fastmerge(DataMat, w = NULL, tol = 1e-04)
```

Arguments

`DataMat`	a numeric matrix/vector with rows as data points.
`w`	an optional numeric vector of the same length as x; Defaults to all elements 1.
`tol`	a numeric value providing the tolerance for identifying duplicates with respect to the first column.

Details

If two values in the first column of DataMat are separated by a value less than tol then the corresponding rows are merged.

Value

A list including the elements

`DataMat`	a numeric matrix/vector with rows sorted with respect to the first column.
`w`	obtained weights corresponding to the merged points.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

See Also

See also the function `smooth.spline`.

Examples

```
args(fastmerge)
x <- runif(100,-1,1)
y <- runif(100,-1,1)
DataMat <- cbind(x, y)
tmp <- fastmerge(DataMat)
```

penta *C code for solving pentadiagonal linear equations.*

Description

This function is only intended for an internal use.

Usage

```
penta(dim, E, A, D, C, F, B, X)
```

Arguments

dim	vector containing dimension of linear system.
E	Internal vector storing for one of the sub-diagonals.
A	Internal vector storing for one of the sub-diagonals.
D	Internal vector storing for one of the sub-diagonals.
C	Internal vector storing for one of the sub-diagonals.
F	Internal vector storing for one of the sub-diagonals.
B	Internal vector storing for the right hand side of linear equation.
X	Vector to store the solution.

Value

Does not return anything. Changes the inputs according to the algorithm.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

predcvxpen *C code for prediction using cvx.lse.reg, cvx.lip.reg and cvx.lse.con.reg for function and its derivatives.*

Description

This function is only intended for an internal use.

Usage

```
predcvxpen(dim, x, t, zhat, deriv, L, U, fun, P, Q, R)
```

Arguments

dim	vector of sample size, size of newdata.
x	Newdata.
t	x-vector in cvx.pen.reg
zhat	prediction obtained from cvx.pen.reg
deriv	derivative vector obtained from cvx.pen.reg
L	Internal vector obtained from cpen function.
U	Internal vector obtained from cpen function.
fun	vector containing the function estimate.
P	Internal vector set to zero.
Q	Internal vector set to zero.
R	Internal vector set to zero.

Details

The estimate is characterized by a fixed point equation which gives the algorithm for prediction.

Value

Does not return anything. Changes the inputs according to the algorithm.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

---

sim.est *Single Index Model Estimation: Objective Function Approach.*

Description

This function provides an estimate of the non-parametric function and the index vector by minimizing an objective function specified by the method argument.

Usage

```
sim.est(x, y, w = NULL, beta.init = NULL, nmulti = NULL, L = NULL,
                  lambda = NULL, maxit = 100, bin.tol = 1e-05, beta.tol = 1e-05,
                    method = c("cvx.pen","cvx.lip","cvx.lse","smooth.pen"),
                    progress = TRUE, force = FALSE)
## Default S3 method:
sim.est(x, y, w = NULL, beta.init = NULL, nmulti = NULL, L = NULL,
                  lambda = NULL, maxit = 100, bin.tol = 1e-05, beta.tol = 1e-05,
                    method = c("cvx.pen","cvx.lip","cvx.lse","smooth.pen"),
                    progress = TRUE, force = FALSE)
```

```
## S3 method for class 'sim.est'
plot(x,...)
## S3 method for class 'sim.est'
print(x,...)
## S3 method for class 'sim.est'
predict(object, newdata = NULL, deriv = 0, ...)
```

Arguments

x	a numeric matrix giving the values of the predictor variables or covariates. For functions plot and print, 'x' is an object of class 'sim.est'.
y	a numeric vector giving the values of the response variable.
method	a string indicating which method to use for regression.
lambda	a numeric value giving the penalty value for cvx.pen and cvx.lip.
L	a numeric value giving the Lipschitz bound for cvx.lip.
w	an optional numeric vector of the same length as x; Defaults to all 1.
beta.init	An numeric vector giving the initial value for the index vector.
nmulti	An integer giving the number of multiple starts to be used for iterative algorithm. If beta.init is provided then the nmulti is set to 1.
bin.tol	A tolerance level upto which the x values used in regression are recognized as distinct values.
beta.tol	A tolerance level for stopping iterative algorithm for the index vector.
maxit	An integer specifying the maximum number of iterations for each initial β vector.
progress	A logical denoting if progress of the algorithm is to be printed. Defaults to TRUE.
force	A logical indicating the use of cvx.lse.reg or cvx.lse.con.reg. Defaults to FALSE and uses cvx.lse.con.reg
object	An object of class 'sim.est'.
...	Any additional arguments to be passed.
newdata	a matrix of new data points in the predict function.
deriv	a numeric either 0 or 1 representing which derivative to evaluate.

Details

The function minimizes

$$\sum_{i=1}^{n} w_i(y_i - f(x_i^\top \beta))^2 + \lambda \int \{f''(x)\}^2 dx$$

with constraints on f dictated by method = 'cvx.pen' or 'smooth.pen'. For method = 'cvx.lip' or 'cvx.lse', the function minimizes

$$\sum_{i=1}^{n} w_i(y_i - f(x_i^\top \beta))^2$$

with constraints on f disctated by method = 'cvx.lip' or 'cvx.lse'. The penalty parameter λ is not choosen by any criteria. It has to be specified for using method $=$ 'cvx.pen', 'cvx.lip' or 'smooth.pen' and λ denotes the Lipschitz constant for using the method $=$ 'cvx.lip.reg'. `plot` function provides the scatterplot along with fitted curve; it also includes some diagnostic plots for residuals and progression of the algorithm. Predict function now allows calculation of the first derivative. In applications, it might be advantageous to scale of the covariate matrix `x` before passing into the function which brings more stability to the algorithm.

Value

An object of class 'sim.est', basically a list including the elements

`beta`	A numeric vector storing the estimate of the index vector.
`nmulti`	Number of multistarts used.
`x.mat`	the input 'x' matrix with possibly aggregated rows.
`BetaInit`	a matrix storing the initial vectors taken or given for the index parameter.
`lambda`	Given input `lambda`.
`L`	Given input `L`.
`K`	an integer storing the row index of `BetaInit` which lead to the estimator `beta`.
`BetaPath`	a list containing the paths taken by each initial index vector for nmulti times.
`ObjValPath`	a matrix with nmulti rows storing the path of objective function value for multiple starts.
`convergence`	a numeric storing convergence status for the index parameter.
`itervec`	a vector of length nmulti storing the number of iterations taken by each of the multiple starts.
`iter`	a numeric giving the total number of iterations taken.
`method`	method given as input.
`regress`	An output of the regression function used needed for predict.
`x.values`	sorted 'x.betahat' values obtained by the algorithm.
`y.values`	corresponding 'y' values in input.
`fit.values`	corresponding fit values of same length as that of $x\beta$.
`deriv`	corresponding values of the derivative of same length as that of $x\beta$.
`residuals`	residuals obtained from the fit.
`minvalue`	minimum value of the objective function attained.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu

Source

Kuchibhotla, A. K., Patra, R. K. and Sen, B. (2015+). On Single Index Models with Convex Link.

Examples

```
args(sim.est)
x <- matrix(runif(50*3,-1,1),ncol = 3)
b0 <- rep_len(1,3)/sqrt(3)
y <- (x%*%b0)^2 + rnorm(50,0,0.3)
tmp1 <- sim.est(x, y, lambda = 0.01, method = "cvx.pen", nmulti = 5)
tmp3 <- sim.est(x, y, lambda = 0.01, method = "smooth.pen", nmulti = 5)
print(tmp1)
print(tmp3)
plot(tmp1)
plot(tmp3)
predict(tmp1, newdata = c(0,0,0))
predict(tmp3, newdata = c(0,0,0))
```

---

simestgcv *Single Index Model Estimation: Objective Function Approach.*

Description

This function provides an estimate of the non-parametric function and the index vector by minimizing an objective function specified by the method argument and also by choosing tuning parameter using GCV.

Usage

```
simestgcv(x, y, w = NULL, beta.init = NULL, nmulti = NULL,
                lambda = NULL, maxit = 100, bin.tol = 1e-06,
                beta.tol = 1e-05, agcv.iter = 100, progress = TRUE)

## Default S3 method:
simestgcv(x, y, w = NULL, beta.init = NULL, nmulti = NULL,
            lambda = NULL, maxit = 100, bin.tol = 1e-06,
            beta.tol = 1e-05, agcv.iter = 100, progress = TRUE)
```

Arguments

x	a numeric matrix giving the values of the predictor variables or covariates. For functions plot and print, 'x' is an object of class 'sim.est'.
y	a numeric vector giving the values of the response variable.
lambda	a numeric vector giving lower and upper bounds for penalty used in cvx.pen and cvx.lip.
w	an optional numeric vector of the same length as x; Defaults to all 1.
beta.init	An numeric vector giving the initial value for the index vector.
nmulti	An integer giving the number of multiple starts to be used for iterative algorithm. If beta.init is provided then the nmulti is set to 1.

agcv.iter	An integer providing the number of random numbers to be used in estimating GCV. See smooth.pen.reg for more details.
progress	A logical denoting if progress of the algorithm to be printed. Defaults to TRUE.
bin.tol	A tolerance level upto which the x values used in regression are recognized as distinct values.
beta.tol	A tolerance level for stopping iterative algorithm for the index vector.
maxit	An integer specifying the maximum number of iterations for each initial β vector.

Details

The function minimizes

$$\sum_{i=1}^{n} w_i(y_i - f(x_i^\top \beta))^2 + \lambda \int \{f''(x)\}^2 dx$$

with no constraints on f. The penalty parameter λ is choosen by the GCV criterion between the bounds given by lambda. Plot and predict function work as in the case of sim.est function.

Value

An object of class 'sim.est', basically a list including the elements

beta	A numeric vector storing the estimate of the index vector.
nmulti	Number of multistarts used.
x.mat	the input 'x' matrix with possibly aggregated rows.
BetaInit	a matrix storing the initial vectors taken or given for the index parameter.
lambda	Given input lambda.
K	an integer storing the row index of BetaInit which lead to the estimator beta.
BetaPath	a list containing the paths taken by each initial index vector for nmulti times.
ObjValPath	a matrix with nmulti rows storing the path of objective function value for multiple starts.
convergence	a numeric storing convergence status for the index parameter.
itervec	a vector of length nmulti storing the number of iterations taken by each of the multiple starts.
iter	a numeric giving the total number of iterations taken.
method	method is always set to "smooth.pen.reg".
regress	An output of the regression function used needed for predict.
x.values	sorted 'x.betahat' values obtained by the algorithm.
y.values	corresponding 'y' values in input.
fit.values	corresponding fit values of same length as that of $x\beta$.
deriv	corresponding values of the derivative of same length as that of $x\beta$.
residuals	residuals obtained from the fit.
minvalue	minimum value of the objective function attained.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu

Source

Kuchibhotla, A. K., Patra, R. K. and Sen, B. (2015+). On Single Index Models with Convex Link.

Examples

```
args(sim.est)
x <- matrix(runif(20*2,-1,1),ncol = 2)
b0 <- rep_len(1,2)/sqrt(2)
y <- (x%*%b0)^2 + rnorm(20,0,0.3)
tmp2 <- simestgcv(x, y, lambda = c(20^{1/6}, 20^{1/4}), nmulti = 1,
     agcv.iter = 10, maxit = 10, beta.tol = 1e-03)
print(tmp2)
plot(tmp2)
predict(tmp2, newdata = c(0,0))
```

---

smooth.pen.reg *Penalized Smooth/Smoothing Spline Regression.*

Description

This function provides an estimate of the non-parameteric regression function using smoothing splines.

Usage

```
smooth.pen.reg(x, y, lambda, w = NULL, agcv = FALSE, agcv.iter = 100, ...)
## Default S3 method:
smooth.pen.reg(x, y, lambda, w = NULL, agcv = FALSE, agcv.iter = 100, ...)
## S3 method for class 'smooth.pen.reg'
plot(x,...)
## S3 method for class 'smooth.pen.reg'
print(x,...)
## S3 method for class 'smooth.pen.reg'
predict(object, newdata = NULL, deriv = 0, ...)
```

Arguments

x	a numeric vector giving the values of the predictor variable. For functions plot and print, 'x' is an object of class 'smooth.pen.reg'.
y	a numeric vector giving the values of the response variable.
lambda	a numeric value giving the penalty value.
w	an optional numeric vector of the same length as x; Defaults to all 1.
agcv	a logical denoting if an estimate of generalized cross-validation is needed.

agcv.iter	a numeric denoting the number of random vectors used to estimate the GCV. See details.
...	additional arguments.
object	An object of class 'smooth.pen.reg'.
newdata	a matrix of new data points in the predict function.
deriv	a numeric either 0 or 1 representing which derivative to evaluate.

Details

The function minimizes

$$\sum_{i=1}^{n} w_i(y_i - f(x_i))^2 + \lambda \int \{f''(x)\}^2 dx$$

without any constraint on f. This function implements in R the algorithm noted in Green and Silverman (1994). The function smooth.spline in R is not suitable for single index model estimation as it chooses λ using GCV by default. plot function provides the scatterplot along with fitted curve; it also includes some diagnostic plots for residuals. Predict function now allows computation of the first derivative. Calculation of generalized cross-validation requires the computation of diagonal elements of the hat matrix involved which is cumbersone and is computationally expensive (and also is unstable). smooth.Pspline of pspline package provides the GCV criterion value which matches the usual GCV when all the weights are equal to 1 but is not clear what it is for weights unequal. We use an estimate of GCV (formula of which is given in Green and Silverman (1994)) proposed by Girard which is very stable and computationally cheap. For more details about this randomized GCV, see Girard (1989).

Value

An object of class 'smooth.pen.reg', basically a list including the elements

x.values	sorted 'x' values provided as input.
y.values	corresponding 'y' values in input.
fit.values	corresponding fit values of same length as that of 'x.values'.
deriv	corresponding values of the derivative of same length as that of 'x.values'.
iter	Always set to 1.
residuals	residuals obtained from the fit.
minvalue	minimum value of the objective function attained.
convergence	Always set to 0.
agcv.score	Asymptotic GCV approximation. Proposed in Silverman (1982) as a computationally fast approximation to GCV.
splinefun	An object of class 'smooth.spline' needed for predict.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

Source

Green, P. J. and Silverman, B. W. (1994) Non-parametric Regression and Generalized Linear Models: A Roughness Penalty Approach. Chapman and Hall.

Girard, D. A. (1989) A Fast ' Monte-Carlo Cross-Validation' Procedure for Large Least Squares Problems with Noisy Data. Numerische Mathematik, 56, 1-23.

Examples

```
args(smooth.pen.reg)
x <- runif(50,-1,1)
y <- x^2 + rnorm(50,0,0.3)
tmp <- smooth.pen.reg(x, y, lambda = 0.01, agcv = TRUE)
print(tmp)
plot(tmp)
predict(tmp, newdata = rnorm(10,0,0.1))
```

---

solve.pentadiag *Pentadiagonal Linear Solver.*

Description

A function to solve pentadiagonal system of linear equations.

Usage

```
## S3 method for class 'pentadiag'
solve(a, b, ...)
```

Arguments

a	a numeric square matrix with pentadiagonal rows. The function does NOT check for pentadiagonal matrix.
b	a numeric vector of the same length as nrows(a). This argument cannot be a matrix.
...	any additional arguments

Details

This function is written mainly for use in this package. It may not be the most efficient code.

Value

A vector containing the solution.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu

Examples

```
A <- matrix(c(2,1,1,0,0,
     1,2,1,1,0,
     1,1,2,1,1,
     0,1,1,2,1,
     0,0,1,1,2),nrow = 5)
b <- rnorm(5)
tmp <- solve.pentadiag(A, b)
```

---

spen_egcv *C code for smoothing splines with randomized GCV computation.*

Description

This function is only intended for an internal use.

Usage

```
spen_egcv(dim, x, y, w, h, QtyPerm, lambda, m, nforApp,
   EGCVflag, agcv)
```

Arguments

dim	vector of sample size.
x	x-vector in smooth.pen.reg.
y	y-vector in smooth.pen.reg.
w	w-vector in smooth.pen.reg.
h	difference vector for x for internal use.
QtyPerm	Second order difference for x for internal use.
lambda	smoothing parameter input for smooth.pen.reg.
m	vector to store the prediction vector.
nforApp	Number of iterations for approximate GCV.
EGCVflag	Logical when GCV is needed.
agcv	Internal scalar. Set to 0. Stores the approximate GCV.

Details

This is same as smooth.spline except for small changes.

Value

Does not return anything. Changes the inputs according to the iterations.

Author(s)

Arun Kumar Kuchibhotla, arunku@wharton.upenn.edu.

See Also

smooth.spline

Index