Package 'simest'

May 9, 2026

Title Constrained Single Index Model Estimation

Type Package

Version 0.4-1-1

Date 2025-11-28

Imports nnls, cobs, stats, graphics

Description Estimation of function and index vector in single index model ('sim') with (and w/o) shape constraints including different smoothness conditions. See, e.g., Kuchibhotla and Patra (2020) <doi:10.3150/19-BEJ1183>.

License GPL-2

BugReports https://r-forge.r-project.org/tracker/?group_id=846

URL https://curves-etc.r-forge.r-project.org/,
https://r-forge.r-project.org/R/?group_id=846,
https://r-forge.r-project.org/scm/viewvc.php/pkg/simest/?root=curves-etc,
svn://svn.r-forge.r-project.org/svnroot/curves-etc/pkg/simest

LazyLoad yes

NeedsCompilation yes

Author Arun Kumar Kuchibhotla [aut] (ORCID:
<https://orcid.org/0000-0003-4459-5352>),
Rohit Kumar Patra [aut],
Martin Maechler [ctb, cre] (fix *.Rd, etc to get back on CRAN, ORCID:
<https://orcid.org/0000-0002-8685-9910>)

Maintainer Martin Maechler <maechler@stat.math.ethz.ch>

Repository CRAN

Date/Publication 2025-12-03 21:10:02 UTC

Contents

---

cpen *Callable 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

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, ...)

## S3 method for class 'cvx.lip.reg'
plot(x, diagnostics = TRUE, ylab = quote(y ~ "and" ~ hat(y) ~ " values"),
   main = sprintf("Convex Lipschitz Regression\n using Least Squares, L=%g", x$L),
     pch = "*", cex = 1, lwd = 2, col2 = "red", ablty = 4, ...)
## S3 method for class 'cvx.lip.reg'
print(x, digits = getOption("digits"), ...)
## 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 `t`; Defaults to all elements $1/n$.
`L`	a numeric value providing the Lipschitz bound on the function.
`...`	additional arguments.
`diagnostics`	for the `plot()` method; if true, as by default, produce diagnostic, notably residual plots additionally.

`main`, `ylab`, `pch`, `cex`, `lwd`, `col2`, `ablty`
: further optional argument to the `plot()` method; the last two for the color and line type of *some* plot components.

`digits`
: the number of significant digits, for numbers in the `print()` method.

`x`, `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 permuted accordingly such that (t_i, z_i) stay pairs.

This function uses the `nnls` function from the **nnls** package to perform the constrained minimization of least squares. The `plot` method provides the scatterplot along with fitted curve; when `diagnostics = TRUE`, it also includes some diagnostic plots for residuals.

The `predict()` method allows calculating the first derivative as well.

Value

An object of class `"cvx.lip.reg"`, basically a list with 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, here.

`convergence`
: a numeric indicating the convergence status of the code.

Author(s)

Arun Kumar Kuchibhotla

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

Function `nnls` from CRAN package **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

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.conreg(t, z, w = NULL, ...)
cvx.lse.con.reg(t, z, w = NULL, ...) # will be deprecated
```

Arguments

`t, z`	numeric vectors (of the same lengths) with the values of the predictor and response variable.
`w`	an optional numeric vector of the same length as `t`; defaults to all equal weights.
`...`	additional arguments, passed to `conreg()`.

Details

This function does the same thing as `cvx.lse.reg` except that here we use the `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

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

```
str(cvx.lse.conreg)
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,...)
## S3 method for class 'cvx.lse.reg'
plot(x, diagnostics = TRUE,
     ylab = quote(y ~ "and" ~ hat(y) ~ " values"),
     pch = "*", cex = 1, lwd = 2, col2 = "red", ablty = 4, ...)
## S3 method for class 'cvx.lse.reg'
print(x, digits = getOption("digits"), ...)
## 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.
`diagnostics`	for the `plot()` method; if true, as by default, produce diagnostics, notably residual plots additionally.
`ylab`, `pch`, `cex`, `lwd`, `col2`, `ablty`	further optional argument to the `plot()` method; the last two for the color and line type of *some* plot components.
`digits`	the number of significant digits, for numbers in the `print()` method.
`x`, `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 permuted accordingly such that (t_i, z_i) stay pairs.

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 package **nnls** for the same purpose.

The `plot` method provides a scatterplot along with the fitted curve; it also includes some diagnostic plots for residuals. The `predict()` method 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

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)
cvxL <- cvx.lse.reg(x, y)
print(cvxL)
plot(cvxL)
predict(cvxL, 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-5, maxit = 1000)

## S3 method for class 'cvx.pen.reg'
plot(x, ylab = quote(y ~ "and" ~ hat(y) ~ " values"),
   main = sprintf("Convex Regression using\n Penalized Least Squares (lambda=%.4g)",
                  x$lambda),
     pch = "*", cex = 1, lwd = 2, col2 = "red", ablty = 4, ...)
## S3 method for class 'cvx.pen.reg'
print(x, digits = getOption("digits"), ...)
## 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 is an object of class "cvx.pen.reg".
y	a numeric vector giving the values of the response variable.
lambda	a nonnegative number, the penalty value aka 'smoothing parameter'.
w	an optional numeric vector of the same length as x; defaults to all 1.
tol	non-negative number, the tolerance level for convergence.
maxit	an integer giving the maxmimum number of steps taken by the algorithm; defaults to 1000.
ylab, main, pch, cex, lwd, col2, ablty	further optional argument to the plot() method; the last two for the color and line type of *some* plot components.
...	any additional arguments, passed, e.g., to lower level plotting functions.
digits	the number of significant digits, for numbers in the print() method.
object	an object of class "cvx.pen.reg"; for the predict() method.
newdata	a vector of new data points to be used in predict().

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

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 functions in R. This function implements this procedure with a given tolerance level.

Usage

```
fastmerge(DataMat, w = NULL, tol = 1e-4)
```

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 DataMat[,1].

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 and possibly merged with respect to the first column.
w	obtained weights corresponding to the merged points.

Author(s)

Arun Kumar Kuchibhotla; also the authors of smooth.spline.

See Also

The function smooth.spline also uses such pre-binning.

Examples

```
 ## relevant example % found in ../tests/fastmerge-ex.R
n <- 47
set.seed(2657) # <- found after searching
x <- sort(signif(runif(n, -1,1), 5))
y <- sinpi(3*x) * exp(-x) + rnorm(n)/10
str(fmL <- fastmerge(cbind(x,y))) # only 44 (out of 47) "unique" x[]
d.fm <- data.frame(fmL)
d2 <- data.frame(fastmerge(cbind(x,y), tol = 25e-4)) # larger tol ==> only 42 "unique"
table(w <- d2$w) # 3x w=2  and  1 w=3
stopifnot(nrow(d.fm) == 44, nrow(d2) == 42,
          identical(    w[w > 1], c(2, 2, 2, 3)),
          identical(which(w > 1), c(5L, 26L, 28L, 39L)),
          all.equal(1000 * fmL$AddVar[fmL$w != 1],
                    c(2.28919, 23.918, 17.5813), tolerance = 3e-6))

plot(y ~ x, type = "b")
lines(d.fm[,1], d.fm[,2], col = adjustcolor(2, 1/2), lwd=3)
lines(d2  [,1], d2  [,2], col = adjustcolor(4, 1/2), lwd=2)
abline(v = d.fm[d.fm$w > 1, 1],  col = 2, lwd=3, lty=2)
abline(v = (xw <- d2[w > 1, 1]), col = 4, lwd=2, lty=3)
axis(3, at= xw, labels=paste("w=",w[w > 1]), col = 4, col.axis = 4)
```

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 of cvx.lse.reg, cvx.lip.reg and cvx.lse.con.reg, function and 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

---

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

Description

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-5, beta.tol = 1e-5,
       method = c("cvx.pen", "cvx.lip", "cvx.lse.con", "cvx.lse", "smooth.pen"),
        progress = TRUE, force = FALSE)

## S3 method for class 'sim.est'
plot(x, pch = 20, cex = 1, lwd = 2, col2 = "red", ...)
## S3 method for class 'sim.est'
print(x, digits = getOption("digits"), ...)
## 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 plot() and print() methods, x is an object of class sim.est.
y	a numeric vector giving the values of the response variable (length as x).
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.
L	a numeric value giving the Lipschitz bound for cvx.lip.
lambda	a numeric value giving the penalty value for cvx.pen and cvx.lip.
maxit	an integer specifying the maximum number of iterations for each initial β vector.
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.
method	a string indicating which method to use for regression.
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. This is *deprecated*; rather method = "cvx.lse.con" or = "cvx.lse" choose the method.
object	the result of sim.est(), of class sim.est.
pch, cex, lwd, col2	further optional arguments to plot() method, passed to underlying plot() or lines() calls.
digits	the number of significant digits, for numbers in the print() method.
...	additional arguments to be passed.
newdata	a matrix of new data points in the predict() method.
deriv	either 0 or 1, the order of the 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 dictated 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"`.

The `plot()` method provides the scatterplot along with the fitted curve; it also includes some diagnostic plots for residuals and progression of the algorithm. The `predict()` method 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 the same length).
`residuals`	residuals obtained from the fit.
`minvalue`	minimum value of the objective function attained.

Author(s)

Arun Kumar Kuchibhotla

References

Arun K. Kuchibhotla and Rohit K. Patra (2020) Efficient estimation in single index models through smoothing splines, *Bernoulli* **26**(2), 1587–1618. doi:10.3150/19BEJ1183

Examples

```
set.seed(2017)
x <- matrix(runif(50*3, -1,1), ncol = 3)
b0 <- c(1, 1, 1)/sqrt(3)
y <- (x %*% b0)^2 + rnorm(50,0,0.3)
(mCP  <- sim.est(x, y, lambda = 0.01, method = "cvx.pen",   nmulti = 5))
(mCLi <- sim.est(x, y, L = 10,        method = "cvx.lip",   nmulti = 3))
(mSP  <- sim.est(x, y, lambda = 0.01, method = "smooth.pen",nmulti = 5))
(mCLs <- sim.est(x, y,                method = "cvx.lse",   nmulti = 1))
## Compare the 4 models on the same data point:
pr000 <- sapply(list(mCP, mCLi, mSP, mCLs), predict, newdata = c(0,0,0))
pr000 # values close to 0
```

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

Description

Estimate 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-6,
          beta.tol = 1e-5, agcv.iter = 100, progress = TRUE)
```

Arguments

`x`	a numeric matrix giving the values of the predictor variables or covariates. For the `plot()` and `print()` methods, `x` is an object of class `sim.est` (as for `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

Source

Arun K. Kuchibhotla and Rohit K. Patra (2020) Efficient estimation in single index models through smoothing splines, *Bernoulli* **26**(2), 1587–1618. doi:10.3150/19BEJ1183

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

Examples

```
x <- matrix(runif(20*2, -1,1), ncol = 2) # 20 x 2
b0 <- rep_len(1,2)/sqrt(2)
y <- (x%*%b0)^2 + rnorm(20,0,0.3)
simG <- simestgcv(x, y, lambda = c(20^(1/6), 20^(1/4)), nmulti = 1,
                  agcv.iter = 10, maxit = 10, beta.tol = 1e-3)
simG # print() method
plot(simG)
predict(simG, newdata = local({ x <- seq(-1.125, 1.125, by = 1/32); cbind(x,x) }))
```

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

Description

Estimate the non-parameteric regression function using smoothing splines.

Usage

```
smooth.pen.reg(x, y, lambda, w = NULL, agcv = FALSE, agcv.iter = 100, ...)

## S3 method for class 'smooth.pen.reg'
plot(x, diagnostics = TRUE,
     ylab = quote(y ~ "and" ~ hat(y) ~ " values"),
     pch = "*", cex = 1, lwd = 2, col2 = "red", ablty = 4, ...)
## S3 method for class 'smooth.pen.reg'
print(x, digits = getOption("digits"), ...)
## 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 plot and print methods, `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, passed further, e.g., to `matplot` or `plot.default` for the `plot()` method.
`diagnostics`	for the `plot()` method; if true, as by default, produce diagnostics, notably residual plots additionally.

`ylab`, `pch`, `cex`, `lwd`, `col2`, `ablty`
: further optional argument to the `plot()` method; the last two for the color and line type of *some* plot components.

`digits`
: the number of significant digits, for numbers in the `print()` method.

`object`
: the result of `smooth.pen.reg()`, of class `smooth.pen.reg`.

`newdata`
: a matrix of new data points for the `predict` method.

`deriv`
: either 0 or 1, the order of the 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()` in the **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

References

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 + 0.3 * rnorm(50)
smP <- smooth.pen.reg(x, y, lambda = 0.01, agcv = TRUE)
smP # -> print() method
plot(smP)
predict(smP, newdata = rnorm(10, 0,0.1))
```

---

solve.pentadiag *Pentadiagonal Linear Solver*

Description

A function to solve pentadiagonal system of linear equations.

Usage

```
solve_pentadiag(a, b)
## method  solve.pentadiag(a, b, ...)  is __deprecated__ now (Nov.18 2025)
```

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.

Details

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

Originally it was documented (and declared) as an S3 method for `solve()`, and class `pentadiagonal` and as function named `solve.pentadiag`, but such classed objects were never used.

Value

A vector containing the solution.

Author(s)

Arun Kumar Kuchibhotla

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 <- 1:5
tools::assertWarning(tmp <- solve.pentadiag(A, b), verbose=TRUE) # deprecated
tmp # 0.5 0.75 -0.75 0.75 2.5
stopifnot( identical(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