Package ‘liso’

February 20, 2015

Type Package
Title Fitting lasso penalised additive isotone models
Version 0.2
Date 2010-05-24
Author Zhou Fang
Maintainer Zhou Fang <fang@stats.ox.ac.uk>
Description Fits lasso (total variation) penalised additive isotone models
Depends MASS, Iso
License GPL
Repository CRAN
Date/Publication 2011-11-17 11:25:59
NeedsCompilation no

R topics documented:

  cv.liso ................................................................. 2
  liso ................................................................. 3
  liso.backfit ...................................................... 5
  liso.covweights ................................................ 6
  liso.maxlamb .................................................... 8
  multistep .......................................................... 9
  plot.multistep .................................................. 10
  predict.multistep ............................................... 12
  print.lisofit ................................................... 13
  seq.log ........................................................... 14
  summary.multistep ............................................. 15

Index 17
**cv.liso**

### Liso Crossvalidation

**Description**

Applies crossvalidation to Liso

**Usage**

```r
cv.liso(x, y, K = 10, lambda = NULL, trace = FALSE, plot.it = FALSE, se = TRUE, weights = rep(1, length(y)), plotCV(cv.object, se=TRUE, ...)
```

**Arguments**

For `cv.liso`:
- `x`: Design matrix (without intercept).
- `y`: Response value.
- `K`: Number of CV folds.
- `lambda`: Values of the penalty parameter lambda to be tried. For speed, it’s advised that a decreasing vector be used. If NULL, a log grid used, using `liso.maxlamb` to calculate the maximum.
- `trace`: If TRUE, print diagnostic information as calculation is done.
- `plot.it`: If TRUE, plot a graph of CV error against lambda with `plotCV`.
- `weights`: Observation weights. Should be a vector of length equal to the number of observations.
- `weightedcv`: If TRUE, use observation weights when averaging CV error across folds.
- `huber`: If less than Inf, huberisation parameter for huberised liso. (Experimental)
- `covweights`: Covariate weights. Should be a vector of length equal to the number of covariates.
- `gridsize`: Size of logarithmic grid of lambda values, if lambda is unspecified.
- `gridmin`: Minimum of logarithmic grid of lambda values, if lambda is unspecified. Considered as a proportion of the maximum value of lambda. For plotCV:
- `cv.object`: Object to be plotted. For both:
- `se`: If TRUE, add error bars to CV plot.
- `...`: Additional arguments to be passed to liso.backfit or plot

**Value**

`cv.liso` creates a list of 5 components:
- `lambda`: Lambda values used.
- `cv`: Mean or weighted mean CV for each lambda.
- `cv.error`: Sqrt of MLE estimated variance of CV for each lambda.
- `residmat`: Full length(lambda) x K matrix of CV errors.
- `optimlam`: Lambda value that minimises CV error
liso

Author(s)
Zhou Fang

References
Zhou Fang and Nicolai Meinshausen (2009), Liso for High Dimensional Additive Isotonic Regression, available at http://blah.com

See Also
liso.backfit

Examples

## Use the method on a simulated data set

set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(rnorm(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(x[, 1] + x[, 2] + rnorm(n))

## Do CV
CVobj <- cv.liso(x, y, K = 10, plot.it = TRUE)

## Do the actual fit
fitobj <- liso.backfit(x, y, CVobj$optimlam)
plot(fitobj)

liso

Automatically conducts Liso fits

Description
An automatic CV and fitting wrapper for Liso.

Usage

liso(x, y, adaptive = TRUE, lambda = NULL, monotone = NULL, control = list(cv = NULL, liso = NULL))

Arguments

x
Design matrix (without intercept).
y
Response value.
adaptive
If TRUE, conduct an adaptive liso type procedure. Otherwise just do the raw liso fits.
lambda Value of the penalty parameter lambda. Default is NULL, specifying repeated cross-validations. Can be a vector, in which case each term gives the lambda for each step of the adaptive procedure.

monotone Monotonicity pattern. Default is NULL, specifying a sign-discovery procedure, or non-monotone fitting if adaptive is FALSE.

control Optional additional arguments to be passed to the cross-validation or backfitting, as a two field list. Each of control$cv, control$liso should be themselves a list, to be passed on as arguments to the relevant part of the procedure.

Details
This function is a convenient wrapper for the liso functions that automates the process of CV and fitting or adaptive fitting.

Value
A lisofit object is returned to represent the fit, which inherits from class multistep. plot, summary, print, `*` and other methods exist.

Author(s)
Zhou Fang

References

See Also
liso.backfit, cv.liso

Examples
```r
## Use the method on a simulated data set
set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]^3 + rnorm(n)

## Do a single prespecified fit
fit1 = liso(x,y, FALSE, 4, TRUE)
plot(fit1, dims=1:2)

## Do a cross-validated fit constrained to be monotone increasing
fit2 = liso(x,y, FALSE, monotone=TRUE)
plot(fit2, dims=1:2)

## Do an adaptive fit constrained to be monotone increasing, with an increased tolerance for convergence in the cross
```
liso.backfit

Function to fit penalized additive isotonic models

Description
Fits penalized additive isotonic models using a total variation penalty.

Usage
liso.backfit(x, y, lambda=0, givebeta = FALSE, tol.target = 1e-04, weights= rep(1, length(y)), covweights, feed, trace = FALSE, monotone, randomise, huber = Inf)

Arguments
- x: Design matrix (without intercept).
- y: Response value.
- lambda: Value of the penalty parameter lambda. Can be either a single value or a vector, in which case the calculations are done sequentially, using the previous calculation as the feed input.
- givebeta: If TRUE, output result as a vector instead of a multistep object.
- tol.target: Threshold at which Liso loss change is considered small enough for convergence.
- weights: Observation weights. Should be a vector of length equal to the number of observations.
- covweights: Covariate weights. Should be a vector of length equal to the number of covariates, or more if different weights are to be applied to positive and negative fits of non-monotone components.
- feed: Initial values for backfitting calculation. By default, the zero fit is used. Any multistep fit can be used instead.
- trace: If TRUE, print diagnostic information as calculation is done.
- monotone: Monotonicity pattern. Can be a single value, or a vector of length equal to the number of covariates. Takes values -1, 0, 1, indicating monotonically decreasing, non-monotonic, monotonically increasing respectively.
- randomise: If TRUE, randomly permute the order of backfitting in each cycle. Usually slower, but possibly more stable.
- huber: If less than Inf, huberization parameter for huberized liso. (Experimental)
Value

With a single value of lambda, a lisofit object is returned, which inherits from class multistep. With more than one value, a list of lisofit values are generated. `plot`, `summary`, `print`, `*` and other methods exist.

Author(s)

Zhou Fang

References


See Also

cv.liso

Examples

```r
## Use the method on a simulated data set

set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1]< 0), scale=FALSE) + x[,2]^3 + rnorm(n)

## Try lambda = 2, lambda = 1
fits <- liso.backfit(x,y, c(2,1), monotone=c(-1,rep(1, 49)))

## plot the result for lambda = 2
plot(fits[[2]])

## Plot y-yhat plot
plot(y,fits[[2]] * x)
```

---

**liso.covweights**

*Covariate Weights for Adaptive Liso*

**Description**

Calculates covariate weights for the Adaptive Liso

**Usage**

`liso.covweights(obj, signfind = FALSE)`
### liso.covweights

**Arguments**

- `obj` Initial fit to use, a multistep object.
- `signfind` If TRUE, conduct monotonicity detection procedure.

**Details**

This function calculates automatically weights for a second run of the Liso algorithm, in an adaptive liso scheme. See example for practical usage.

**Value**

Produces a vector of covariate weights to be supplied as the covweight argument in liso.backfit.

**Author(s)**

Zhou Fang

**References**


**Examples**

```r
## Use the method on a simulated data set

set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]^3 + rnorm(n)

## Adaptive liso
initialfit = liso.backfit(x, y, 4)
secondfit = liso.backfit(x, y, 4, covweights = liso.covweights(initialfit))

## Compare sparsity
which(dim(initialfit) != 0)
which(dim(secondfit) != 0)

set.seed(79)
y2 <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]^3-6*(abs(x[,2] - 1) < 0.1) + rnorm(n)

## Sign finding
initialfit = liso.backfit(x, y2, 2, monotone=FALSE)
secondfit = liso.backfit(x, y2, 2, monotone=FALSE, covweights = liso.covweights(initialfit, signfind=TRUE))

## Compare monotonicity. Note near x=1
plot(secondfit, dim=2)
plot(initialfit, dim=2, add=TRUE, col=2)
```
**liso.maxlam**

**Description**

Calculates maximum value of lambda for which Liso gives a non-zero fit.

**Usage**

```r
liso.maxlam(x=NULL, y=NULL, monotone=TRUE, covweights=rep(1, ncol(x)), weights=rep(1, length(y)))
```

**Arguments**

- `x`: Design matrix (without intercept).
- `y`: Response value.
- `monotone`: Monotonicity pattern. Can be a single value, or a vector of length equal to the number of covariates. Takes values -1, 0, 1, indicating monotonically decreasing, non-monotonic, monotonically increasing respectively.
- `covweights`: Covariate weights. Should be a vector of length equal to the number of covariates, or more if different weights are to be applied to positive and negative fits of non-monotone components.
- `weights`: Observation weights. Should be a vector of length equal to the number of observations.

**Author(s)**

Zhou Fang

**References**


**See Also**

`plotCV`
Examples

```r
## Use the method on a simulated data set

set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]*3 + rnorm(n)

liso.maxlamb(x, y)
```

---

**multistep**  
*Multidimensional step functions*

**Description**

Produces a multistep object

**Usage**

```r
multistep(coefchain,x=NULL,intercept=0,sortedx = apply(x,2,sort),names = NULL, pinters=NULL,...)
```

**Arguments**

- `coefchain`  
  Vector of step sizes at each observation point for each vector, concatenated as a single vector.

- `x`  
  Matrix of observations `coefchain` corresponds to.

- `intercept`  
  Intercept value. i.e. value of mean(f(x)).

- `sortedx`  
  x sorted in each column.

- `names`  
  Names to be assigned to covariates.

- `pinters`  
  The values of the component functions at the left ends of each range.

- `...`  
  Additional variables to be stored in the final object.

**Details**

This function generates a multistep object, to represent a function that is the sum of right-continuous step functions on each input. Internally, the function is stored in a sparse format.

sortedx and pinters are calculated, if not provided.

Multistep objects may be plotted. They may also be evaluated at a particular vector value, or matrix of values, through the * operator or the predict function.
plot.multistep

Value

Produces a multistep object.

Author(s)

Zhou Fang

See Also

plot.multistep, summary.multistep, predict.multistep

Examples

## Produces a 2d step function

```r
set.seed(79)
n <- 100; p <- 2

## Pick some random knots
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
obj = multistep(rep(0.1, (n-1)*p), x)
x2 <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
obj * x2 - obj*x
image(outer(-50:50/10, -50:50/10, function(x,y) obj*c(x,y)))
```

Description

Produces covariate plots for a multidimensional step function.

Usage

```r
## S3 method for class 'multistep'
plot(x = NULL, xpoints=NULL, ypoints = NULL, dims = 1:max(nrow(x$param), ncol(xpoints)), ylimit = cbind(min(min(x$y), max(x$y)), max(max(x$y), min(x$y))), grid = TRUE, add = FALSE, titles = NA)
```

Arguments

- `x`: A multistep object.
- `xpoints`: Covariate values of additional points to be plotted.
- `ypoints`: Response values of additional points to be plotted.
- `dims`: Dimensions to be shown. (Default is all)
- `ylimit`: Y-axis limits to be used for all plots.
- `grid`: If TRUE, construct a grid of plots to show all plotted components. Otherwise, plot each component after the other normally.
add If TRUE, superimpose new plot on the old plot. This may false for more than one component.

titles If TRUE, add names of covariates to plot.

Value

If grid is TRUE, return the old par() values before function was called.

Author(s)

Zhou Fang

References

Zhou Fang and Nicolai Meinshausen (2009), Liso for High Dimensional Additive Isotonic Regression, available at http://blah.com

See Also

multistep, plot

Examples

## Use the method on a simulated data set
set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]^3 + rnorm(n)

## try lambda = 2
fits <- liso.backfit(x, y, 2)
fits2 <- liso.backfit(x, y, 4)

## Plot in some different ways
plot(fits, dim=2)
plot(fits2, dim=2, col=2, add=TRUE)

plot(fits, grid=FALSE)
plot(fits)
**predict.multistep**  
*Multidimensional step function evaluation*

**Description**

Evaluates a multistep type function at a given value.

**Usage**

```
## S3 method for class 'multistep'
predict(object, newx, ...)
## S3 method for class 'multistep'
e1 * e2
```

**Arguments**

- `object`: A multistep object.
- `newx`: Values to evaluate the represented function at. Each row is considered to be a separate observation.
- `...`: Additional arguments for compatibility.
- `e1`: Either a multistep object or a matrix to evaluate it at.
- `e2`: Either a multistep object or a matrix to evaluate it at. One of `e1`, `e2` must be a matrix, or vector.

**Value**

Produces a vector of results.

**Note**

`predict(object, newx)` is equivalent to `object * newx`, which is also equivalent to `newx * object`.

**Author(s)**

Zhou Fang

**See Also**

`multistep`
print.lisofit

Examples

## Produces a 2d step function

```r
set.seed(79)
n <- 100; p <- 2

## Choose some random knots
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
obj = multistep(rep(0.1, (n-1)*p), x)
x2 <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
predict(obj,x) - obj*x
```

print.lisofit  
*Prints details for a liso fit*

Description

Prints information and diagnostic statistics for a particular Liso fit.

Usage

## S3 method for class 'lisofit'

```r
print(x, ...)  
```

Arguments

- `x`  
  A lisofit object. Dummy variables for compatibility:

- `...`  
  Unused.

Details

print prints, in this case, n, p, Lambda for the fit, and then for each non-zero fitted variable, stepwise and total variation complexity statistics, as well as the apparent monotonicity of the fit if it was not pre-specified. Finally some residual statistics are printed.

Author(s)

Zhou Fang

References


See Also

- multistep, summary.multistep
Examples

```r
## Use the method on a simulated data set
set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]^3 + rnorm(n)

## try lambda = 2
fits <- liso.backfit(x, y, 2)
print(fits)
```

Description

Generates a log grid

Usage

`seq.log(from = 1, to = 1, length.out = 50, add.zero = FALSE, shifting = 0, ...)`

Arguments

- `from`: First value.
- `to`: Final value.
- `length.out`: Number of values to generate.
- `add.zero`: If TRUE, append the value 0 on the smaller end of the result.
- `shifting`: Shifting to apply to the log grid. Negative values produce greater bunching up near the minimum, with the reverse for positive values. NOTE: `shifting + from` and `shifting + to` must both be greater than zero.
- `...`: Dummy variables for compatibility

Value

A vector of length `length.out`, plus one if `add.zero` is TRUE.

Author(s)

Zhou Fang

Examples

```r
seq.log(1, 10, 10)
seq.log(1, 10, 10, FALSE, -0.9)
```
Summary multistep objects

Description

Calculates a variety of summary statistics for multistep (multidimensional step function) objects.

Usage

```r
## S3 method for class 'multistep'
max(x, ..., na.rm)
## S3 method for class 'multistep'
min(x, ..., na.rm)
## S3 method for class 'multistep'
dim(x)
## S3 method for class 'multistep'
abs(x)
## S3 method for class 'multistep'
summary(object, ...)
```

Arguments

- `x`: A multistep object.
- `object`: A multistep object. Dummy variables for compatibility:
  - `...`: Unused.
  - `na.rm`: Unused.

Details

- 'max' and 'min' returns the maximum or minimum respectively of each covariate component.
- 'dim' returns the number of non-zero steps in each covariate component.
- 'abs' returns the total variation of each covariate component.
- 'summary' returns a list containing all of the above.

Value

- For 'max', 'min', 'abs', 'dim', a vector with length equal to the number of covariates.
- For 'summary', a list containing 'max', 'min', 'totalvar', 'dim', each being a vector of length equal to the number of covariates.

Author(s)

Zhou Fang
References

Zhou Fang and Nicolai Meinshausen (2009), Liso for High Dimensional Additive Isotonic Regression, available at http://blah.com

See Also

multistep

Examples

## Use the method on a simulated data set
set.seed(79)
n <- 100; p <- 50

## Simulate design matrix and response
x <- matrix(runif(n * p, min = -2.5, max = 2.5), nrow = n, ncol = p)
y <- scale(3 * (x[,1] > 0), scale=FALSE) + x[,2]^3 + rnorm(n)

## try lambda = 2
fits <- liso.backfit(x,y, 2)

## Plot some diagnostics
plot(max(fits))
plot(min(fits))
plot(abs(fits))
plot(dim(fits))
print(summary(fits))
Index

*Topic models
  cv.liso, 2
  liso, 3
  liso.backfit, 5
  liso.covweights, 6
  liso.maxlamb, 8
  print.lisofit, 13
  summary.multistep, 15
*Topic regression
  liso, 3
  liso.backfit, 5
  print.lisofit, 13
  summary.multistep, 15
*.multistep (predict.multistep), 12
abs.multistep (summary.multistep), 15

  cv.liso, 2, 4, 6

  dim.multistep (summary.multistep), 15

  liso, 3
  liso.backfit, 3, 4, 5
  liso.covweights, 6
  liso.maxlamb, 8

  max.multistep (summary.multistep), 15
  min.multistep (summary.multistep), 15
  multistep, 9, 11–13, 16

  plot, 11
  plot.multistep, 10, 10
  plotCV, 8
  plotCV (cv.liso), 2
  predict.multistep, 10, 12
  print.lisofit, 13

  seq.log, 14
  summary.multistep, 10, 13, 15