Package ‘orderedLasso’

February 20, 2015

Title Ordered Lasso and Time-lag Sparse Regression
Version 1.7
Author Jerome Friedman, Xiaotong Suo and Robert Tibshirani
Description Ordered lasso and time-lag sparse regression. Ordered Lasso fits a
linear model and imposes an order constraint on the coefficients. It writes
the coefficients as positive and negative parts, and requires positive
parts and negative parts are non-increasing and positive. Time-Lag Lasso
generalizes the ordered Lasso to a general data matrix with multiple
predictors. For more details, see Suo, X., Tibshirani, R., (2014) 'An
Ordered Lasso and Sparse Time-lagged Regression'.
Maintainer Xiaotong Suo <xiaotong@stanford.edu>
Depends R (>= 3.0.0), Matrix
Imports Iso, quadprog, ggplot2, reshape2
License GPL-2
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-11-27 08:32:37

R topics documented:

orderedLasso ......................................................... 2
orderedLasso.cv ...................................................... 3
orderedLasso.path .................................................... 4
predict.orderedLasso ................................................. 6
predict.orderedLasso.path .......................................... 6
predict.timeLagLasso ............................................... 7
predict.timeLagLasso.path .......................................... 8
timeLagLasso ........................................................ 8
timeLagLasso.cv ..................................................... 10
timeLagLasso.path ................................................... 12
time_lag_matrix ....................................................... 13

Index 14
orderedLasso  

Fit an ordered lasso

Description

One of the main functions. Ordered Lasso builds a regression model with an L1-constraint imposed on the coefficients. The coefficients are re-written as negative and positive parts and the model requires positive and negative parts are monotone non-increasing and positive.

Usage

orderedLasso(x, y, lambda, intercept = TRUE, b0 = NULL, beta_pos = NULL, beta_neg = NULL, method = c("solve.QP", "GG"), strongly.ordered = FALSE, standardize = TRUE, trace = FALSE, niter = 500, iter.gg = 100, epsilon = 1e-08)

Arguments

x  A matrix of predictors, where the rows are the samples and the columns are the predictors
y  A vector of observations, where length(y) equals nrow(x)
lambda  Regularization parameter(>0)
intercept  TRUE if there is an intercept in the model.
b0  Initial value for the intercept.
beta_pos  Optional vector of initialization of the positive part of the coefficients.
beta_neg  Optional vector of initialization of the negative part of the coefficients.
method  Two options available, Solve.QP and Generalized Gradient. Solve.QP uses the package quadprog to solve a quadratic programming problem. GG stands for generalized gradient. GG uses proximal generalized gradient algorithm to solve the problem. Detailed can be seen in the paper refered in the description.
strongly.ordered  An option which allows users to order the coefficients in absolute value. The coefficients returned by the orderedLasso may not be monotone non-increasing in absolute value even though the positive parts and negative parts are monotone non-increasing. Details can be seen in the paper refered in the Description. Strongly.ordered options returns the coefficients monotone non-increasing in absolute value.
standardize  Standardize the data matrix x
trace  Output option; trace = TRUE gives verbose output
niter  Maximum number of iterations; default 500.
iter.gg  Number of iterations of generalized gradient; default 100
epsilon  Error tolerance parameter for convergence criterion ; default 1e-05.
Value

bp  Estimated coefficients- positive part
bn  Estimated coefficients- negative part
beta Estimated coefficients, which are equal to bp - bn
b0  Estimated intercept, if there is one in the model.
fitted Fitted values of y
type  Type of model fit, "gaussian"
call The call to orderedLasso

Examples

set.seed(3)
n = 50
b = c(7,3,1,0)
p = length(b)
x = matrix(rnorm(n*p), nrow = n)
sigma = 4
y = x %*% b + sigma * rnorm(n, 0, 1)
result1 = orderedLasso(x, y, lambda = 1, intercept = TRUE, standardize = TRUE,
method = "GG", strongly.ordered = TRUE)
result2 = orderedLasso(x, y, lambda = 1, intercept = TRUE, standardize = TRUE,
strongly.ordered = TRUE)
print(result1)
print(result2)

orderedLasso.cv  Cross-validation function for the ordered lasso

Description

Uses cross-validation to estimate the regularization parameter for the ordered lasso model.

Usage

orderedLasso.cv(x, y, lamlist = NULL, minlam = NULL, maxlam = NULL,
nlam = 50, flmin = 5e-04, strongly.ordered = FALSE, intercept = TRUE,
standardize = TRUE, nfolds = 10, folds = NULL, niter = 500,
iter.gg = 100, method = c("Solve.QP", "GG"), trace = FALSE,
epsilon = 1e-05)

Arguments

x A matrix of predictors, where the rows are the samples and the columns are the predictors
y A vector of observations, where length(y) equals nrow(x)
orderedLasso.path

Description

Fit a path of ordered lasso models over different values of the regularization parameter.

Usage

```r
orderedLasso.path(x, y, lamlist = NULL, minlam = NULL, maxlam = NULL, nlam = 50, flmin = 0.005, intercept = TRUE, standardize = TRUE, method = c("Solve.QP", "GG"), niter = 500, iter.gg = 100, strongly.ordered = FALSE, trace = FALSE, epsilon = 1e-05)
```
orderedLasso.path

Arguments

- **x**: A matrix of predictors, where the rows are the samples and the columns are the predictors.
- **y**: A vector of observations, where length(y) equals nrow(x).
- **lamlist**: Optional vector of values of lambda (the regularization parameter).
- **minlam**: Optional minimum value for lambda.
- **maxlam**: Optional maximum value for lambda.
- **nlam**: Number of values of lambda to be tried. Default nlam = 50.
- **flmin**: Fraction of maxlam; minlam= flmin*maxlam. If computation is slow, try increasing flmin to focus on the sparser part of the path.
- **intercept**: True if there is an intercept in the model.
- **standardize**: Standardize the data matrix x. Default is TRUE.
- **method**: Two options available, Solve.QP and Generalized Gradient.
- **niter**: Number of iterations of ordered lasso, initialized to 500.
- **iter.gg**: Number of iterations of generalized gradient; Default iter.gg = 100.
- **strongly.ordered**: An option which allows users to order the coefficients non-decreasing in absolute value. Details can be seen in the orderedLasso Description.
- **trace**: Output option; trace=TRUE gives verbose output.
- **epsilon**: Error tolerance parameter for convergence criterion. Default is 1e-5.

Value

- **bp**: p by nlam matrix of estimated positive coefficients (p=#variables).
- **bn**: p by nlam matrix of estimated negative coefficients.
- **beta**: p by nlam matrix of estimated coefficients.
- **b0**: a length nlam vector of estimated intercepts.
- **lamlist**: Vector of values of lambda used.
- **err**: Vector of errors.
- **call**: The call to orderedLasso.path.

Examples

```r
set.seed(3)
n = 50
b = c(4,3,1,0)
p = length(b)
x = matrix(rnorm(n*p), nrow = n)
sigma = 5
y = x %*% b + sigma * rnorm(n, 0, 1)
path1 = orderedLasso.path(x, y, intercept = FALSE,
                          method = "Solve.QP", strongly.ordered = TRUE)
plot(path1)
print(path1)
```
predict.orderedLasso  
*make predictions from a fitted "orderedLasso" object*

**Description**

Similar to other predict methods, this functions predicts fitted values from a fitted "orderedLasso" object.

**Usage**

```r
## S3 method for class 'orderedLasso'
predict(object, newdata, ...)```

**Arguments**

- `object`  
  fitted "orderedLasso" model
- `newdata`  
  Matrix of new values for object at which predictions are to be made.
- `...`  
  Not used. Other arguments to predict.

**Value**

- `yhat`  
  fitted y values

---

predict.orderedLasso.path  
*make predictions from a fitted "orderedLasso.path" object*

**Description**

Similar to other predict methods, this functions predicts fitted values from a fitted "orderedLasso" object.

**Usage**

```r
## S3 method for class 'orderedLasso.path'
predict(object, newdata, s = NULL,
         exact = FALSE, ...)```

**Arguments**

- `object`  
  fitted "orderedLasso" model
- `newdata`  
  Matrix of new values for object at which predictions are to be made.
- `s`  
  Not used. Other arguments to predict.
**predict.timeLagLasso**

**Arguments**

- **object**: fitted "orderedLasso.path" model
- **newdata**: Matrix of new values for object at which predictions are to be made.
- **s**: Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
- **exact**: If exact = TRUE, and predictions are to made at values of s not included in the original fit, these values of s are merged with object$lambda, and the model is refit before predictions are made. If exact = FALSE (default), then the predict function uses linear interpolation to make predictions for values of s that do not coincide with those used in the fitting algorithm. Note that exact = TRUE is fragile when used inside a nested sequence of function calls. predict.orderedLasso() needs to update the model, and expects the data used to create it to be around.
- **...**: Not used. Other arguments to predict.

**Value**

- **nfit**: A matrix or a vector of fitted values

---

**predict.timeLagLasso**  
*make predictions from a fitted "timeLagLasso" object*

**Description**

Similar to other predict methods, this function predicts fitted values from a fitted "timeLagLasso" object.

**Usage**

```r
## S3 method for class 'timeLagLasso'
predict(object, newdata = NULL, ...)
```

**Arguments**

- **object**: fitted "timeLagLasso" model
- **newdata**: Matrix of new values for object at which predictions are to be made.
- **...**: Not used. Other arguments to predict.

**Value**

- **yhat**: fitted y values
- **yhat.ordered**: fitted y values of the strongly ordered coefficients
predict.timeLagLasso.path

make predictions from a fitted "timeLagLasso.path" object

Description

Similar to other predict methods, this function predicts fitted values from a fitted "timeLagLasso.path" object.

Usage

## S3 method for class 'timeLagLasso.path'
predict(object, newdata = NULL, s = NULL, 
         exact = FALSE, ...)

Arguments

- **object**: fitted "timeLagLasso.path" model
- **newdata**: Matrix of new values for object at which predictions are to be made.
- **s**: Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
- **exact**: If exact = TRUE, and predictions are to made at values of s not included in the original fit, these values of s are merged with object$lambda, and the model is refit before predictions are made. If exact = FALSE (default), then the predict function uses linear interpolation to make predictions for values of s that do not coincide with those used in the fitting algorithm. Note that exact = TRUE is fragile when used inside a nested sequence of function calls. predict.orderedLasso() needs to update the model, and expects the data used to create it to be around.
- **...**: Not used. Other arguments to predict.

Value

- **yhat**: A matrix or a vector of fitted values
- **yhat.ordered**: A matrix or a vector of fitted values of the strongly ordered lasso coefficients

timeLagLasso

Fit a time-lag lasso

Description

Fit a time-lag lasso model. Builds a regression model with multiple predictors, where an ordered constraint is imposed on each predictor.
timeLagLasso

Usage

timeLagLasso(x, y, lambda, maxlag, intercept = TRUE, standardize = TRUE,
  beta_pos = NULL, beta_neg = NULL, b0 = NULL, strongly.ordered = TRUE,
  maxiter = 500, inneriter = 100, iter.gg = 100, method = c("Solve.QP",
  ", "GG"), trace = FALSE, epsilon = 1e-05)

Arguments

  x          A matrix of predictors, where the rows are the samples and the columns are the
             predictors.
  y          A vector of observations, where length(y) equals nrow(x).
  lambda     Regularization parameter(>0).
  maxlag     maximum time-lag specified by user.
  intercept  TRUE if there is an intercept in the model. center=FALSE should almost never
             be used. This option is available for special uses only.
  standardize Standardize the data matrix x.
  beta_pos   Optional vector of initialization of the positive part of the coefficients.
  beta_neg   Optional vector of initialization of the negative part of the coefficients.
  b0         Initialization of the intercept.
  strongly.ordered
             An option which allows users to order each predictor coefficients non-increasing
             in absolute value. Details can be seen in the orderedLasso description; default
             TRUE.
  maxiter    Maximum iterations run by time-lag lasso. Initialized to 500.
  inneriter  Maximum iterations run by orderedLasso. Initialized to 100.
  iter.gg    Maximum iterations run by generalized gradient. Initialized to 100
  method     Two options available, Solve.QP and Generalized Gradient
  trace      Output option; trace = TRUE gives verbose output.
  epsilon    Error tolerance parameter for convergence criterion; default 1e-5.

Details

  First transfer the data matrix x to the correct format corresponding to the maxlag specified by user.
  Then use coordinate descent method by calling orderedLasso to get updates for each predictor.

Value

  bp          Estimated coefficients of the positive part
  bn          Estimated coefficients of the negative part
  beta        Estimated coefficients of predictors, which are equal to bp - bn
  b0          Estimated intercept if there is one in the model
  fitted      Fitted values of y
  maxlag      Maximum time-lag
  p           The number of predictors
  type        Type of model fit, "gaussian" in this case
Examples

```r
set.seed(3)
n = 50
maxlag = 5
num_rows_needed = n + maxlag + 1
sigma = 4
x = matrix(rnorm(num_rows_needed * 4), nrow = num_rows_needed)
x_new = time_lag_matrix(x, maxlag)
b = c(3, 1, 0, 0,
     4, 1, 0, 0,
     2, 1, 0, 0,
     1, 0, 0, 0)
y = x_new %*% b + sigma * rnorm(nrow(x_new))
y = as.vector(y)
y = c(y, rnorm(maxlag + 4))
model1 = timeLagLasso(x = x, y = y, lambda = 1, maxlag = maxlag,
                      method = "solve.QP", strongly.ordered = TRUE)
```

Description

Uses cross-validation to estimate the regularization parameter for timeLagLasso model

Usage

```r
timeLagLasso.cv(x, y, maxlag, lamlist = NULL, minlam = NULL,
                 maxlam = NULL, nlam = 10, flmin = 0.01, flmax = 1, intercept = TRUE,
                 standardize = TRUE, method = c("solve.QP", "GG"),
                 strongly.ordered = TRUE, nfolds = 10, folds = NULL, maxiter = 500,
                 inneriter = 100, iter.gg = 100, trace = FALSE, epsilon = 1e-04)
```

Arguments

- `x`: A matrix of predictors, where the rows are the samples and the columns are the predictors
- `y`: A vector of observations, where length(y) equals nrow(x)
- `maxlag`: Maximum time-lag variable chosen by user
- `lamlist`: Optional vector of values of lambda (the regularization parameter)
- `minlam`: Optional minimum value for lambda
- `maxlam`: Optional maximum value for lambda
- `nlam`: Number of values of lambda to be tried
- `flmin`: Fraction of maxlam minlam= flmin*maxlam. If computation is slow, try increasing flmin to focus on the sparser part of the path; default 1e-2
flmax  Multiplication of maxlam \( \text{maxlam} = \text{flmax} \times \text{maxlam} \); default 1
intercept  True if there is an intercept in the model.
standardize  Standardize the data matrix \( x \).
method  Two options available, Solve.QP and Generalized Gradient. Details about two options can be seen in the orderedLasso description.

strongly.ordered  An option which allows users to order the coefficients in absolute value.
nfolds  Number of cross-validation folds
folds  (Optional) user-supplied cross-validation folds. If provided, nfolds is ignored.
maxiter  maximum iterations run by time-lag lasso. Initialized to 500.
inneriter  maximum iterations run by orderedLasso. Initialized to 100.
iter.gg  Maximum iterations run by generalized gradient. Initialed to 100
trace  Output option; trace = TRUE gives verbose output.
epsilon  Error tolerance parameter for convergence criterion; default 1e-5

Value

lamlist  Vector of lambda values tried
cv.err  Estimate of cross-validation error
cv.se  Estimated standard error of cross-validation estimate
lamhat  lambda value minimizing cv.err
folds  Indices of folds used in cross-validation
lamhat.1se  largest lambda value with cv.err less than or equal to min(cv.err)+ SE
nonzero  Vector giving number of non-zero coefficients for each lambda value
call  The call to timeLagLasso.cv

Examples

```r
set.seed(3)
n = 50
maxlag = 5
num_rows_needed = n + maxlag + 1
sigma = 4
x = matrix(rnorm(num_rows_needed * 4), nrow = num_rows_needed)
x_new = time_lag_matrix(x, maxlag)
b = c(3,1,1,0,0,
    4,1,0,0,0,
    3,2,1,0,0,
    1,0,0,0,0)
y = x_new %*% b + sigma* rnorm(nrow(x_new))
y = as.vector(y)
y = c(y, rnorm(maxlag + 1))
cvmodel = timeLagLasso.cv(x= x, y = y, maxlag = 5, method = "Solve.QP")
```
timeLagLasso.path 

Fit a path of time-lasso models

Description
Fit a path of time-lasso models

Usage

```
timeLagLasso.path(x, y, lamlist = NULL, minlam = NULL, maxlam = NULL, nlam = 10, flmin = 0.01, strongly.ordered = TRUE, flmax = 1, maxlag, intercept = TRUE, standardize = TRUE, method = c("solve.QP", "GG"), maxiter = 500, inneriter = 100, iter.gg = 100, trace = FALSE, epsilon = 1e-05)
```

Arguments

- **x**: A matrix of predictors, where the rows are the samples and the columns are the predictors
- **y**: A vector of observations, where length(y) equals nrow(x)
- **lamlist**: Optional vector of values of lambda (the regularization parameter)
- **minlam**: Optional minimum value for lambda
- **maxlam**: Optional maximum value for lambda
- **nlam**: Number of values of lambda to be tried
- **flmin**: Fraction of maxlam minlam= flmin*maxlam. If computation is slow, try increasing flmin to focus on the sparser part of the path; default = 1e-2.
- **strongly.ordered**: An option which allows users to order the coefficients in absolute value.
- **flmax**: Multiplication of maxlam maxlam = flmax * maxlam. Default = 1
- **maxlag**: Maximum time-lag chosen by user.
- **intercept**: True if there is an intercept in the model.
- **standardize**: Standardize the data matrix x.
- **method**: Two options available, Solve.QP and Generalized Gradient
- **maxiter**: Maximum iterations run by time-lag lasso. Initialied to 500.
- **inneriter**: maximum iterations run by orderedLasso. Initialized to 100.
- **iter.gg**: Maximum iterations run by generalized gradient. Initialized to 100
- **trace**: Output option; trace = TRUE gives verbose output.
- **epsilon**: Error tolerance parameter for convergence criterion; default 1e-5
Value

bp | p by nlam matrix of estimated positive coefficients (p=#variables)
bn | p by nlam matrix of estimated negative coefficients
beta | p by nlam matrix of estimated coefficients
b0 | a vector of length nlam of estimated intercept
lamlist | Vector of values of lambda used
err | Vector of errors
maxlag | Maximum time-lag variable
p | The number of predictors
fitted | a length(y) by nlam matrix of fitted values
call | The call to "timeLagLasso.path"

Examples

set.seed(3)
n = 50
maxlag = 5
num_rows_needed = n + maxlag + 1
sigma = 4
x = matrix(rnorm(num_rows_needed * 4), nrow = num_rows_needed)
x_new = time_lag_matrix(x, maxlag)
b = c(3,1,1,0,0,
    4,1,0,0,0,
    3,2,1,0,0,
    1,0,0,0,0)
y = x_new %*% b + sigma* rnorm(nrow(x_new))
y = as.vector(y)
y = c(y, rnorm(maxlag + 1))
path1 = timeLagLasso.path(x= x, y = y, maxlag = 5, method = "Solve.QP", strongly.ordered = TRUE)
plot(path1)

time_lag_matrix

time_lag_matrix

description

Build the time lag matrix for the data matrix x.

Usage

time_lag_matrix(x, maxlag)

Arguments

x | A matrix of predictors, where the rows are the samples and the columns are the predictors
maxlag | maximum time lag variable.
Index

orderedLasso, 2
orderedLasso.cv, 3
orderedLasso.path, 4

predict.orderedLasso, 6
predict.orderedLasso.path, 6
predict.timeLagLasso, 7
predict.timeLagLasso.path, 8

time_lag_matrix, 13
timeLagLasso, 8
timeLagLasso.cv, 10
timeLagLasso.path, 12