Package ‘Coxnet’

December 10, 2015

Type Package
Title Regularized Cox Model
Version 0.2
Date 2015-12-09
Author Xiang Li, Donglin Zeng and Yuanjia Wang
Maintainer Xiang Li <xlRT7S@columbia.edu>
Description Cox model regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty, and their adaptive forms, such as adaptive lasso and net adjusting for signs of linked coefficients. Moreover, it treats the number of non-zero coefficients as another tuning parameter and simultaneously selects with the regularization parameter \( \lambda \). In addition, it fits a varying coefficient Cox model by kernel smoothing, incorporated with the aforementioned penalties. The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients.
License GPL (>= 2)
Imports Rcpp (>= 0.11.5)
LinkingTo Rcpp, RcppEigen
Depends Matrix (>= 1.1-5)
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-12-10 00:04:05

R topics documented:

Coxnet-package .......................................................... 2
Coxnet, loCoxnet ........................................................... 3
coxsplit ................................................................. 7
print.Coxnet ............................................................ 8

Index 10
Regularized Cox Model

Description

This package fits a Cox model regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty, and their adaptive forms, such as adaptive lasso and net adjusting for signs of linked coefficients. Moreover, it treats the number of non-zero coefficients as another tuning parameter and simultaneously selects with the regularization parameter $\lambda$.

In addition, it fits a varying coefficient Cox model by kernel smoothing, incorporated with the aforementioned penalties.

The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients.

Details

Package: Coxnet
Type: Package
Version: 0.2
Date: 2015-12-09
License: GPL (>= 2)

Functions: Coxnet, loCoxnet, print.Coxnet, coxsplit

Author(s)

Xiang Li, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xl2473@columbia.edu>

References

http://www.jstatsoft.org/v33/i01/

http://www.jstatsoft.org/v39/i05/


Examples

```r
set.seed(1213)
N=100; p=30; p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=0.3,size=1) # censoring indicator
y=cbind(time=ty,status=1-tcens)
fiti=coxnet(x,y,penalty="Lasso") # Lasso
```

Description

Coxnet fits a Cox model regularized with net, elastic-net or lasso penalty, and their adaptive forms, such as adaptive lasso and net adjusting for signs of linked coefficients. Moreover, it treats the number of non-zero coefficients as another tuning parameter and simultaneously selects with the regularization parameter lambda.

loCoxnet fits a varying coefficient Cox model by kernel smoothing, incorporated with the aforementioned penalties.

The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients.

Usage

```r
Coxnet(x, y, Omega = NULL, penalty = c("Lasso", "Enet", "Net"),
  alpha = 1, lambda = NULL, nlambda = 50, rlambda = NULL, nfolds = 1, foldid = NULL,
  inzero = TRUE, adaptive = c(FALSE, TRUE), aini = NULL, isd = FALSE,
  ifast = TRUE, keep.beta = FALSE, thresh = 1e-6, maxit = 1e+5)

loCoxnet(x, y, w, w0 = NULL, h = NULL, hnext = NULL, Omega = NULL,
  penalty = c("Lasso", "Enet", "Net"), alpha = 1, lambda = NULL,
  nlambda = 50, rlambda = NULL, nfolds = 1, foldid = NULL,
  adaptive = c(FALSE, TRUE), aini = NULL, isd = FALSE, keep.beta = FALSE,
  thresh = 1e-6, thresh2 = 1e-10, maxit = 1e+5)
```
Arguments

- **x**: input matrix. Each row is an observation vector.
- **y**: response variable. `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating event, and '0' indicating right censored.
- **w**: input vector, same length as `y`. The coefficients vary with `w`.
- **w0**: evaluation local points. The output of estimates are evaluated at these local value `w0`. If `w0 = NULL`, `w0` is generated as 10 equally spaced points in the range of `w`.
- **h**: bandwidth.
- **hnext**: an increase in bandwidth `h`. Default is 0.01.
- **Omega**: correlation/adjacency matrix with zero diagonal, used for `penalty = "Net"` to calculate Laplacian matrix.
- **penalty**: penalty type. Can choose "Net", "Enet" and "Lasso". For "Net", need to specify `Omega`; otherwise, "Enet" is performed.
- **alpha**: ratio between L1 and Laplacian for "Net", or between L1 and L2 for "Enet". Can be zero and one. For `penalty = "Net"`, the penalty is defined as

\[ \lambda \alpha + (1 - \alpha)/2 * (\beta^T L \beta) \]

where `L` is a Laplacian matrix calculated from `Omega`. For `adaptive = c(FALSE, TRUE)`, its calculation is also based on an initial estimate of \( \beta \) from regularized Cox model with `penalty = "Enet"`, `alpha = 1` to adjust for signs of coefficients. For `penalty = "Enet"`, the penalty is defined as

\[ \lambda \alpha + ||\beta||_1 + (1 - \alpha)/2 * ||\beta||_2 \]

- **lambda**: a user supplied decreasing sequence. If `lambda = NULL`, a sequence of `lambda` is generated based on `nlambda` and `rlambda`. Supplying a value of `lambda` overrides this.
- **nlambda**: number of `lambda` values. Default is 50.
- **rlambda**: fraction of `lambda` to determine the smallest value for `lambda`. The default is `rlambda = 0.0001` when the number of observations is larger than or equal to the number of variables; otherwise, `rlambda = 0.01`.
- **nfolds**: number of folds. Default is `nfolds = 1` and `foldid = NULL` and cross-validation is not performed. For cross-validation, smallest value allowable is `nfolds = 3`. Specifying `foldid` overrides this.
- **foldid**: an optional vector of values between 1 and `nfolds` specifying which fold each observation is in.
- **inzero**: logical flag for simultaneously tuning the number of non-zero coefficients with `lambda`. Default is `inzero = TRUE`.
- **adaptive**: logical flags for adaptive version. Default is `adaptive = c(FALSE, TRUE)`. The first element is for adaptive on `\beta` in L1 and the second for adjusting for signs of linked coefficients in Laplacian matrix.
a user supplied initial estimate of $\beta$. It is a list including \texttt{wbeta} for adaptive L1 and \texttt{sgn} for adjusting Laplacian matrix. \texttt{wbeta} is the absolute value of inverse initial estimates. If aini = NULL but adaptive is required, aini is generated from regularized Cox model with penalty = "Enet", alpha = 1.

\texttt{isd} logical flag for outputing standardized coefficients. \texttt{x} is always standardized prior to fitting the model. Default is \texttt{isd} = FALSE, returning $\beta$ on the original scale.

\texttt{ifast} logical flag for efficient calculation of risk set updates. Default is \texttt{ifast} = TRUE.

\texttt{keep.beta} logical flag for returning estimates for all lambda values. For \texttt{keep.beta} = FALSE, only return the estimate with the largest cross-validation partial likelihood.

\texttt{thresh} convergence threshold for coordinate descent. Default value is $1\times10^{-6}$.

\texttt{thresh2} threshold for removing very small lambda value for local methods. The algorithm computes along a sequence of lambda value until any absolute value of the second derivative is smaller than \texttt{thresh2}. The estimates are reported based on this smaller set of lambda value. Default value is $1\times10^{-10}$.

\texttt{maxit} Maximum number of iterations for coordinate descent. Default is $10^5$.

Details

One-step coordinate descent algorithm is applied for each lambda. \texttt{ifast} = TRUE adopts an efficient way to update risk set and sometimes the algorithm ends before all nlambda values of lambda have been evaluated. To evaluate small values of lambda, use \texttt{ifast} = FALSE. The two methods only affect the efficiency of algorithm, not the estimates.

Cross-validation partial likelihood is used for tuning parameters. For \texttt{inzero} = TRUE, we further select the number of non-zero coefficients obtained from regularized Cox model at each lambda. This is motivated by formulating L0 variable selection in ADMM form.

For vayring coefficients methods, the bandwidth is selected by cross-validation. We recommend to check whether a small increase of $h$, say $h+h_{next}$, will improve the current cvm.

Value

Coxnet outputs an object with S3 class "Coxnet".

- **Beta** estimated coefficients.
- **Beta0** coefficients after tuning the number of non-zeros, for inzero = TRUE.
- **fit** a data.frame containing lambda and the number of non-zero coefficients nzero. For cross-validation, additional results are reported, such as average cross-validation partial likelihood cvm and its standard error cvse, and index with max indicating the largest cvm.
- **fit0** a data.frame containing lambda, cvm and nzero based on inzero = TRUE.
- **lambda.max** value of lambda that gives maximum cvm.
- **lambda.opt** value of lambda based on inzero = TRUE.
- **cv.nzero** cvm with length of number of non-zero components of Beta0. The kth value of cv.nzero corresponds to retaining the k largest non-zero coefficients (absolute values) in Beta0. The optimal number of non-zero is selected by the maximum value of cv.nzero at lambda = lambda.opt.
penalty penalty type.
adaptive logical flags for adaptive version (see above).
flag convergence flag (for internal debugging). flag = 0 means converged.

loCoxnet outputs an object with S3 class "Coxnet" and "loCoxnet".

Beta a list of estimated coefficients with length of lambda. If there are more than one w0 value, each element of the list is a matrix with p rows and the number of columns is the length of w0. If there is one w0, Beta is a matrix rather than a list, with p rows and nlambda columns.
fit a data.frame containing lambda and the number of non-zero coefficients nzero. For cross-validation, additional results are reported, such as average cross-validation partial likelihood cvm and its standard error cvse, and index with max indicating the largest cvm.
lambda.max value of lambda that gives maximum cvm.
cvh a data.frame containing bandwidth, cvm and cvse.
penalty penalty type.
adaptive logical flags for adaptive version (see above).
flag convergence flag (for internal debugging). flag = 0 means converged.

Author(s)
Xiang Li, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xl2473@columbia.edu>

References
http://www.jstatsoft.org/v33/i01/
http://www.jstatsoft.org/v39/i05/

See Also
print.Coxnet, coxsplit
Examples

```r
set.seed(123)
N=100; p=30; pl=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(pl)
xb=x[,1:pl]
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1)  # censoring indicator
y=cbind(time=ty,status=1-tcens)

fiti=Coxnet(x,y,penalty="Lasso",nlambda=10,nfolds=10)  # Lasso
# attributes(fiti)
```

---

**coxsplit**  
*Split Data for Cross-validation*

**Description**

Specify which fold each observation is in to obtain `foldid`.

**Usage**

```r
coxsplit(y, nfolds)
coxsplity(y, nfolds)
coxsplitw(w, nfolds)
```

**Arguments**

- `y`: response variable. `y` should be a two-column matrix with columns named ‘time’ and ‘status’. The latter is a binary variable, with ‘1’ indicating event, and ‘0’ indicating right censored.
- `w`: input vector, same length as `y`. The coefficients vary with `w`.
- `nfolds`: number of folds.

**Details**

By default, coxsplit is incorporated in Coxnet and coxsplitw in loCoxnet. To use other data splitting method, generate `foldid` at the outset and supply it as an augment in Coxnet and loCoxnet.

**Value**

A vector of values between 1 and `nfolds` specifying which fold each observation is in.
print.Coxnet

Author(s)
Xiang Li, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xl2473@columbia.edu>

See Also
Coxnet, loCoxnet, print.Coxnet

Examples
set.seed(1213)
N=100
xb=rnorm(N)
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1) # censoring indicator
y=cbind(time=ty,status=1-tcens)
foldid=coxsplit(y,10) # 10-fold

print.Coxnet                     Print a Coxnet Object

Description
Print a summary of results along the path of lambda.

Usage
## S3 method for class 'Coxnet'
print(x, digits = 4, ...)

Arguments
x          fitted Coxnet object
digits     significant digits in printout
...         additional print arguments

Details
The performed model is printed, followed by fit and fit0 (if any) from a fitted Coxnet object, or
fit and cvh (if any) from a fitted loCoxnet object

Value
The matrix above is silently returned
Author(s)

Xiang Li, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xl2473@columbia.edu>

See Also

Coxnet, loCoxnet, coxspli

Examples

set.seed(123)
N=1000;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1) # censoring indicator
y=cbind(time=ty,status=1-tcens)

fiti=Coxnet(x,y,penalty="Lasso",nlambda=10,nfolds=10) # Lasso
print(fiti)
Index

*Topic **Cox model**
  Coxnet, loCoxnet, 3
  Coxnet-package, 2
coxsplit, 7
  print.Coxnet, 8
*Topic **Print**
  print.Coxnet, 8
*Topic **Regularization**
  Coxnet, loCoxnet, 3
  Coxnet-package, 2
coxsplit, 7
  print.Coxnet, 8
*Topic **Split**
  coxsplit, 7
*Topic **package**
  Coxnet-package, 2

Coxnet, 2, 7–9
Coxnet (Coxnet, loCoxnet), 3
Coxnet, loCoxnet, 3
Coxnet-package, 2
coxsplit, 2, 6, 7, 9
coxsplitw (coxsplit), 7
coxsplity (coxsplit), 7

loCoxnet, 2, 7–9
loCoxnet (Coxnet, loCoxnet), 3

print.Coxnet, 2, 6, 8, 8