Package ‘Coxnet’

March 17, 2015

Type Package

Title Regularized Cox Model

Version 0.1

Date 2015-03-16

Author Xiang Li, Donglin Zeng and Yuanjia Wang

Maintainer Xiang Li <x12473@columbia.edu>

Description Cox model regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty. In addition, it efficiently solves an approximate L0 variable selection based on truncated likelihood function. Moreover, it can also handle the adaptive version of these regularization forms, such as adaptive lasso and net adjusting for signs of linked coefficients. 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 2015-03-17 07:36:38

R topics documented:

Coxnet-package .................................................. 2
Coxnet .............................................................. 3
coxsplit .......................................................... 6
print.Coxnet ..................................................... 7

Index 9
Description

This package fits Cox model regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty. In addition, it efficiently solves an approximate L0 variable selection based on truncated likelihood function. Moreover, it can also handle the adaptive version of these regularization forms, such as adaptive lasso and net adjusting for signs of linked coefficients.

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: 0.1
Date: 2015-03-16
License: GPL (>= 2)

Two functions: `coxnet, print.Coxnet, coxspli

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

set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
Coxnet 3
beta=rnorm(p1)
xb=x[,1:p1]
ty=exp(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")  # Lasso

Coxnet
Fit a Cox Model with Various Regularization Forms

Description
Fit a Cox model regularized with net, elastic-net or lasso penalty. In addition, it efficiently solves an approximate L0 variable selection based on truncated likelihood function (L0-Trunc). Moreover, it can also handle the adaptive version of these regularization forms, such as adaptive lasso and net adjusting for signs of linked coefficients. The regularization path is automatically computed.

Usage
Coxnet(x, y, Omega = NULL, penalty = c("Lasso","Enet", "Net"),
alpha = 1, lambda = NULL, nlambda = 50, rlambda = NULL,
nfolds = 1, foldid = NULL, itrunc = TRUE, adaptive = c(FALSE,TRUE), aini = NULL,
alambda = NULL, nalambda = 10, isd = TRUE, ifast = TRUE, keep.beta = FALSE,
thresh = 1e-7, 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 death, and '0' indicating right censored.
Omega adjancy 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
\[ \lambda \cdot \alpha \cdot \| \beta \|_1 + (1 - \alpha)/2 \cdot (\beta^T L \beta), \]
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
\[ \lambda \cdot \alpha \cdot \| \beta \|_1 + (1 - \alpha)/2 \cdot \| \beta \|_2 \]
lambda: a user supplied decreasing sequence. If lambda = NULL, a sequency of lambda is generated based on nlambda and rlambda.

nlambda: number of lambda values. Default is 50.

rlambda: fraction of lambda.max to determine the smallest value for lambda. The default is $r_{\lambda} = 0.0001$ when the number of observations is larger than or equal to the number of variables; otherwise, $r_{\lambda} = 0.01$.

nfolds: number of folds. Default is 10. Smallest value allowable is nfolds = 3.

foldid: an optional vector of values between 1 and nfolds specifying which fold each observation is in.

itrunc: logical flag for approximate L0 based on truncated likelihood function. Default is itrunc = 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.

aini: a user supplied initial estimate of $\beta$. It is a list including wbeta for adaptive L1 and sgn for adjusting Laplacian matrix. If aini = NULL, aini is generated from regularized Cox model with penalty = "Enet", alpha = 1.

alambda: a user supplied decreasing sequence used in initial estimate. If alambda = NULL, the sequency of alambda is generated based on nlambda and rlambda.

nalambda: number of alambda values. Default is 10.

isd: logical flag for x variable standardization, prior to fitting the model sequence. The coefficients $\beta$ are always returned on the original scale. Default is isd = TRUE.

ifast: logical flag for efficient calculation of risk set updates. Default is ifast = TRUE.

keep.beta: logical flag for returning all the estimates of $\beta$ for each lambda value. For keep.beta = FALSE, only return the estimate with the largest cross-validation partial likelihood and another based on one standard error rule.

thresh: convergence threshold for coordinate descent. Default value is 1E-7.

maxit: Maximum number of iterations for coordinate descent. Default is 10^5.

Details

One-step coordinate descent algorithm is applied for each lambda. 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, ifast = FALSE can be used. The two methods only affect the efficiency of algorithm, not the estimates.

Cross-validation partial likelihood is used for tuning parameters. For itrunc = TRUE, an approximate L0 variable selection based on truncated likelihood function (L0-Trunc) is also performed.

Value

An object with S3 class "Coxnet".

Beta: a sparse Matrix of coefficients, stored in class "dgCMatrix".

Beta0: coefficients based on L0-Trunc, for itrunc = 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 one index with ** and * indicating the largest cvm and one standard error respectively. index = *** implies that the largest cvm is same as the one based on one standard error.

fit0 a data.frame containing lambda, cvm and nzero based on L0-Trunc, for itrunc = TRUE.

lambda.max value of lambda that gives maximum cvm.
lambda.1se value of lambda based on one standard error.
lambda.opt value of lambda based on L0-Trunc.
cv.nzero cvm with length of number of non-zero components of Beta0. The kth value of cv.nzero corresponds to retaining 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).

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
set.seed(1213)
N=100; p=30; p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]
ty=rep(N,exp(xb))
coxsplit

**Split Data for Cross-validation**

**Description**

Specify which fold each observation is in to obtain foldid. coxsplit splits data stratified by 'status' of y. and also based on the 'time' of y. split.foldid splits data stratified by 'status' of y and within each 'status' randomly divides them.

**Usage**

```r
coxsplit(y, nfolds, seeds = NULL)
```

**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 death, and '0' indicating right censored.
- `nfolds` number of folds.
- `seeds` seeds for random split within each 'status' of y. Default is NULL. For seeds=NULL, the split is performed based on the 'time' of y. If seeds is supplied by user, data is randomly divided within each 'status' of y.

**Details**

seeds=NULL is incorporated in Coxnet by default. If random split is needed, foldid can be generated by coxsplit at the outset and input as an augment in Coxnet.

**Value**

a vector of values between 1 and nfolds specifying which fold each observation is in.

**Author(s)**

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

**See Also**

Coxnet, print.Coxnet
print.Coxnet

Examples

```r
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**

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

**Arguments**

- `x`: fitted Coxnet object
- `digits`: significant digits in printout
- `...`: additional print arguments

**Details**

The penalty and whether adaptive is performed is printed, followed by `fit` and then `fit0` (if any), from fitted Coxnet 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`, `coxsplit`
Examples

```r
set.seed(1213)
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, 3
  Coxnet-package, 2
  coxsplit, 6
  print.Coxnet, 7
*Topic **Print**
  print.Coxnet, 7
*Topic **Regularization**
  Coxnet, 3
  Coxnet-package, 2
  coxsplit, 6
  print.Coxnet, 7
*Topic **Split**
  coxsplit, 6
*Topic **package**
  Coxnet-package, 2

  Coxnet, 2, 3, 6, 7
  Coxnet-package, 2
  coxsplit, 2, 5, 6, 7

  print.Coxnet, 2, 5, 6, 7