Package ‘smog’

October 14, 2018

Type Package

Title Structural Modeling by using Overlapped Group Penalty

Version 1.0

Date 2018-10-12

Author Chong Ma

Maintainer Chong Ma <chong.ma@yale.edu>

Description This R package is built for selecting the true predictor variables in the generalized linear model, among a large scale of predictor variables, by following the specficed hierarchical structures. The function glog is implemented by combining the ISTA and ADMM algorithms, and works for continuous, multimonial and survival data.

License GPL (>= 2)

Imports Rcpp (>= 0.12.18), coxed

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

NeedsCompilation yes

Archs i386, x64

R topics documented:

glog — *Generalized linear model constraint on hierarchical structure by using overlapped group penalty*

Description

Generalized linear model constraint on hierarchical structure by using overlapped group penalty

Usage

```
glog(y, x, g, v, hierarchy, lambda, type = "lm", rho = 0.001,
  scale = TRUE, eabs = 0.001, erel = 0.001, LL = 100, eta = 1.25,
  maxitr = 500L)
```

Arguments

`y` a vector of numeric value for response variable in the generalized linear regression. A matrix of n by 2 for survival objects. See `Surv`.

`x` the design matrix of n by p.

`g` a vector of group labels for the p predictor variables.

`v` a vector of 0 and 1 for the penalization status of the p predictor variables. 1 is for penalization and 0 for not penalization.

`hierarchy` hierarchy indicator. 0 for L2 penalty, 1 for the composite L1 and L2 penalty, and 2 for the composite L1, L2 and ridge penalty for each group, respectively.

`lambda` penalty parameters, should correspond to the hierarchy status.

`type` character variable, for different linear models based on the response variable. For continuous response variable, type is set “lm”; for multinomial or binary response variable, type is set “binomial”; for survival response variable, type is set “survival”, respectively.

`rho` The penalty parameter in the ADMM algorithm. Default is 1e-3.

`scale` Whether or not scale the design matrix. Default is true.

`eabs` The absolute tolerance in the ADMM algorithm. Default is 1e-3.

`erel` The reletive tolerance in the ADMM algorithm. Default is 1e-3.

`LL` Initial value for the coefficient of the second-order term in the Majorization-Minimization step.

`eta` gradient step in the FISTA algorithm.

`maxitr` The maximum iterations in the ADMM algorithm. Default is 500.

Examples

```
require(coxed)
n=50;p=1000
set.seed(2018)
# set design matrix
s=10
x=matrix(0,n,1+2*p)
x[,1]=sample(c(0,1),n,replace = TRUE)
```

```
x[,seq(2,1+2*p,2)]=matrix(rnorm(n*p),n,p)
x[,seq(3,1+2*p,2)]=x[,seq(2,1+2*p,2)]*x[,1]

# set beta
beta=c(rnorm(13,0,2),rep(0,ncol(x)-13))
beta[c(2,4,7,9)]=0

# set y
data1=x%*%beta
noise1=rnorm(n)
snr1=as.numeric(sqrt(var(data1)/(s*var(noise1))))
y1=data1+snr1*noise1
g=c(p+1,rep(1:p,rep(2,p)))
v=c(0,rep(1,2*p))
## Not run:
lfit1=glog(y=as.matrix(y1),x=as.matrix(x),g=g,v=v,
          hierarchy=1,lambda=c(0.01,0.001))

## End(Not run)

## binomial data
prob=exp(as.matrix(x)%*%as.matrix(beta))/(1+exp(as.matrix(x)%*%as.matrix(beta)))
y2=ifelse(prob<0.5,0,1)
## Not run:
lfit2=glog(y=as.matrix(y2),x=as.matrix(x),g=g,v=v,
           hierarchy=1,lambda=c(0.025,0.001))

## End(Not run)

## survival data
data3=sim.survdata(N=n,T=100,X=x,beta=beta)
y3=data3$data[,c("y","failed")]
y3$failed=ifelse(y3$failed,1,0)

## Not run:
lfit3=glog(y=as.matrix(y3),x=as.matrix(x),g=g,v=v,
           hierarchy=1,lambda=c(0.075,0.001),
           type="survival")

## End(Not run)
```

iglog — *Integrative generalized linear model constraint on hierarchical structure by using overlapped group penalty*

Description

Integrative generalized linear model constraint on hierarchical structure by using overlapped group penalty

Usage

```
iglog(y1, x1, y2, x2, g, v, hierarchy, lambda, type = "lm", rho = 0.001,
  scale = TRUE, eabs = 0.001, erel = 0.001, LL = 100, eta = 1.25,
  maxitr = 500L)
```

Arguments

`y1` a survival object contains the survival time and censoring status from data1. See `Surv`.

`x1` the design matrix of n by p from data1.

`y2` a survival object contains the survival time and censoring status from data2. See `Surv`.

`x2` the design matrix of n by p from data2. x1 and x2 should have the same number of columns.

`g` a vector of group labels for the p predictor variables.

`v` a vector of 0 and 1 for the penalization status of the p predictor variables. 1 is for penalization and 0 for not penalization.

`hierarchy` hierarchy indicator. 0 for L2 penalty, 1 for the composite L2 penalty, and 2 for the composite L2 and ridge penalty for each group, respectively.

`lambda` penalty parameters, should correspond to the hierarchy status.

`type` character variable, for different linear models based on the response variable. For continuous response variable, type is set “lm”; for multinomial or binary response variable, type is set “binomial”; for survival response variable, type is set “survival”, respectively.

`rho` The penalty parameter in the ADMM algorithm. Default is 1e-3.

`scale` Whether or not scale the design matrix. Default is true.

`eabs` The absolute tolerance in the ADMM algorithm. Default is 1e-3.

`erel` The reletive tolerance in the ADMM algorithm. Default is 1e-3.

`LL` Initial value for the coefficient of the second-order term in the Majorization-Minimization step.

`eta` gradient step in the FISTA algorithm.

`maxitr` The maximum iterations in the ADMM algorithm. Default is 500.

Examples

```
require(coxed)
n=50;p=1000
set.seed(2018)
# generate two design matrices x1 and x2
s=10
x1=matrix(0,n,1+2*p)
x1[,1]=sample(c(0,1),n,replace = TRUE)
x1[,seq(2,1+2*p,2)]=matrix(rnorm(n*p),n,p)
x1[,seq(3,1+2*p,2)]=x1[,seq(2,1+2*p,2)]*x1[,1]

x2=matrix(0,n,1+2*p)
x2[,1]=x1[,1]
x2[,seq(2,1+2*p,2)]=matrix(rnorm(n*p),n,p)
x2[,seq(3,1+2*p,2)]=x2[,seq(2,1+2*p,2)]*x2[,1]

# generate beta1 and beta2
beta1=beta2=c(rnorm(13,0,2),rep(0,ncol(x1)-13))
beta2[1:13]=beta2[1:13]+rnorm(13,0,0.1)
beta1[c(2,4,7,9)]=beta2[c(2,4,7,9)]=0
```

```
# generate two continuous y1 and y2
ldata1=x1%*%beta1
noise1=rnorm(n)
snr1=as.numeric(sqrt(var(ldata1)/(s*var(noise1))))
ly1=ldata1+snr1*noise1

ldata2=x2%*%beta2
noise2=rnorm(n)
snr2=as.numeric(sqrt(var(ldata2)/(s*var(noise1))))
ly2=ldata2+snr2*noise2

g=c(p+1,rep(1:p,rep(2,p)))
v=c(0,rep(1,2*p))
## Not run:
ilfit1=iglog(y1=as.matrix(ly1),x1=as.matrix(x1),
             y2=as.matrix(ly2),x2=as.matrix(x2),
             g=g,v=v,hierarchy=1,lambda=c(0.01,0.001),
             type="lm")

## End(Not run)

## generate two binomial data
prob1=exp(as.matrix(x1)%*%as.matrix(beta1))/(1+exp(as.matrix(x1)%*%as.matrix(beta1)))
cy1=ifelse(prob1<0.5,0,1)

prob2=exp(as.matrix(x2)%*%as.matrix(beta2))/(1+exp(as.matrix(x2)%*%as.matrix(beta2)))
cy2=ifelse(prob2<0.5,0,1)

## Not run:
ilfit2=iglog(y1=as.matrix(cy1),x1=as.matrix(x1),
             y2=as.matrix(cy2),x2=as.matrix(x2),
             g=g,v=v,hierarchy=1,lambda=c(0.025,0.001),
             type="binomial")

## End(Not run)

## generate two survival data
sdata1=sim.survdata(N=n,T=100,X=x1,beta=beta1)
sy1=sdata1$data[,c("y","failed")]
sy1$failed=ifelse(sy1$failed,1,0)

sdata2=sim.survdata(N=n,T=100,X=x2,beta=beta2)
sy2=sdata2$data[,c("y","failed")]
sy2$failed=ifelse(sy2$failed,1,0)

## Not run:
ilfit3=iglog(y1=as.matrix(sy1),x1=as.matrix(x1),
             y2=as.matrix(sy2),x2=as.matrix(x2),
             g=g,v=v,hierarchy=1,lambda=c(0.075,0.001),
             type="survival")

## End(Not run)
```

`penalty` *Penalty function on the composite L1, L2, and ridge penalty*

Description

Penalty function on the composite L1, L2, and ridge penalty

Usage

```
penalty(x, lambda, hierarchy, d)
```

Arguments

`x`	A numeric vector of two.
`lambda`	a vector of three penalty parameters. λ[1] is the L2 penalty for x, λ[2] is the ridge penalty for x, and λ[3] is the L1 penalty for x[2], respectively.
`hierarchy`	Indicator variable for 0, 1, 2. 0 is for no overlap, 1 for composite L1 and L2 penalty, and 2 for composite L1, L2 and ridge penalty, respectively.
`d`	indices for overlapped variables in x.

`prox` *proximal operator on the composite L1, L2, and ridge penalty*

Description

proximal operator on the composite L1, L2, and ridge penalty

Usage

```
prox(x, lambda, hierarchy, d)
```

Arguments

`x`	A numeric vector of two.
`lambda`	a vector of three penalty parameters. λ[1] is the L2 penalty for x, λ[2] is the ridge penalty for x, and λ[3] is the ridge penalty for x[2], respectively.
`hierarchy`	Indicator variable for 0, 1, 2. 0 is for no overlap, 1 for composite L1 and L2 penalty, and 2 for composite L1, L2 and ridge penalty, respectively.
`d`	indices for overlapped variables in x.

proxL1 *proximal operator on L1 penalty*

Description

proximal operator on L1 penalty

Usage

```
proxL1(x, lambda)
```

Arguments

`x`	numeric value.
`lambda`	numeric value for the L1 penalty parameter.

proxL2 *proximal operator on L2 penalty*

Description

proximal operator on L2 penalty

Usage

```
proxL2(x, lambda)
```

Arguments

`x`	A numeric vector.
`lambda`	numeric value for the L2 penalty parameter.

Index