Package ‘ASDAR’

February 9, 2020

Type Package

Title L0 Regularized High-dimensional Accelerated Failure Time Model

Version 1.0

Date 2020-02-07

Author Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang

Maintainer Shuang Zhang <zhangshuang_jz@sina.cn>

Description A constructive approach for L0 penalized estimation in the sparse accelerated failure time (AFT) model with high-dimensional covariates. A computational algorithm that generates a sequence of solutions iteratively, based on active sets derived from primal and dual information and root fnding according to the KKT conditions.

License GPL (>= 2)

Imports Rcpp (>= 1.0.3)

LinkingTo Rcpp, RcppEigen

Archs i386, x64

R topics documented:

`ASDAR-package` *The proposed algorithms of the paper ’L0 Regularized High-dimensional Accelerated Failure Time Model’*

Description

A constructive approach for L0 penalized estimation in the sparse accelerated failure time (AFT) model with high-dimensional covariates. Our proposed method is based on Stute's weighted least squares criterion combined with L0 penalization. This method is a computational algorithm that generates a sequence of solutions iteratively, based on active sets derived from primal and dual information and root finding according to the KKT conditions.

Details

This package contains two mainly algorithm functions, two data generation functions, which are used to provide custom simulation data and a few tool functions. `Asdar` is the method proposed in this paper.

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

References

1. Huang J, Jiao Y, Liu Y, et al. A constructive approach to L0 penalized regression[J]. The Journal of Machine Learning Research, 2018, 19(1): 403-439.
2. Feng X, Huang J, Jiao Y, Zhang S. L0 Regularized High-dimensional Accelerated Failure Time Model.

See Also

Optional links to other man pages

Examples

```
## Not run:
get_data(n, p, beta, varr1, alpha, mu2, varr2, c_r, seed = 1L)
get_weighted_data(n, p, beta, varr1, alpha, mu2, varr2, c_r, seed = 1L)
get_weight(x, y, status)
get_sdar(ita0, T1, x, y, tau1, dd, iter_max)
Asdar(x, y, varr2, ita0, tau, tau1, dd, iter_max)
F_function(ita, x_ita, y_ita)
DF_function(ita, x_ita, y_ita)

## End(Not run)
```

`Asdar` *The AFT-SDAR algorithm proposed in the paper 'L0 Regularized High-dimensional Accelerated Failure Time Model'*

Description

a constructive approach for L0 penalized estimation in the sparse accelerated failure time (AFT) model with high-dimensional covariates.

Usage

```
Asdar(x, y, varr2, ita0, tau, tau1, dd, iter_max)
```

Arguments

x	The motified X, which can be obtained by `get_weighted_data`.
y	The motified Y, which can be obtained by `get_weighted_data`.
varr2	The iteration of finding the best T1 break out when ϵ<varr2^2
ita0	The initial input of η in SDAR algorithm.
tau	The step size of the iteration of finding the best T1.
tau1	The step size 0<τ<1 in the definitions of the active and inactive sets. Default: `tau1=1`
dd	The diagonal element vector of matrix D, $\frac{\sqrt{n}}{\|\tilde{x}_i\|_2}$, i=1,...,p.
iter_max	A maximum number of iterations.

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
data1=get_weighted_data(n,p,beta,varr1,alpha,mu2,varr2,c.r)
tau1=1
iter.max=20
ita0=rep(0,10000)
tau = 20
x.ita=data1[[1]]
y.ita=data1[[2]]
dd = data1[[4]]
res = Asdar(x.ita, y.ita, varr2, ita0, tau, tau1, dd, iter.max)

## End(Not run)
```

DF_function *The first derivative of criterion function.*

Description

The first derivative of criterion function.

$$\bar{X}^T(\bar{Y} - \bar{X}\eta^\diamond)/n,$$

where $\bar{X}$ and $\bar{Y}$ are the motified data,which can be obtained by `get_weighted_data`.

Usage

```
DF_function(ita, x_ita, y_ita)
```

Arguments

`ita`	The estimated value of η
`x_ita`	The motified X, which can be obtained by `get_weighted_data`.
`y_ita`	The motified Y, which can be obtained by `get_weighted_data`.

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
data1=get_weighted_data(n,p,beta,varr1,alpha,mu2,varr2,c.r)
x.ita=data1[[1]]
y.ita=data1[[2]]
ita0=rep(0,10000)
DF_function(ita, x_ita, y_ita)

## End(Not run)
```

F_function *The criterion function in ASDAR.*

Description

The criterion $\mathcal{L}_2(\eta) = \frac{1}{2n} \left\|\bar{Y} - \bar{X}\eta\right\|_2^2$. $\bar{X}$ and $\bar{Y}$ are the motified data, which can be obtained by get_weighted_data.

Usage

```
F_function(ita, x_ita, y_ita)
```

Arguments

ita	The estimated value of η
x_ita	The motified X, which can be obtained by get_weighted_data.
y_ita	The motified Y, which can be obtained by get_weighted_data.

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
data1=get_weighted_data(n,p,beta,varr1,alpha,mu2,varr2,c.r)
x.ita=data1[[1]]
y.ita=data1[[2]]
ita0=rep(0,10000)
F_function(ita, x_ita, y_ita)

## End(Not run)
```

get_data *This function to generate data x, log(T.observe) and status.*

Description

This function generates the simulation data. Refer to the simulation setting in paper 'L0 Regularized High-dimensional Accelerated Failure Time Model'. $\tilde{X}$i.i.d$\sim N(0,1)$,$x_1 = \tilde{x}_1$, $x_p = \tilde{x}_p$ and $x_j = \tilde{x}_j + \alpha(\tilde{x}_{j+1} + \tilde{x}_{j-1})$, j=2,...,p-1 $ln(T_i) = x_i^T\beta + \epsilon_i$

Usage

```
get_data(n, p, beta, varr1, alpha, mu2, varr2, c_r, seed = 1L)
```

Arguments

n	The sample size.
p	The variable dimension.
beta	The underlying regression coeffcient vector β
varr1	The standard error of normal distribution that generates $\tilde{X}$. Default: `varr1=1`
alpha	A measure of the correlation among covariates
mu2	ϵ_i is generated independently from $N(mu2, varr2^2)$. Default: `mu2=0`
varr2	ϵ_i is generated independently from $N(mu2, varr2^2)$
c_r	The censoring rate.
seed	Random seed. Default: `seed=1L`

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
start_time = proc.time()
data1=get_data(n,p,beta,varr1,alpha,mu2,varr2,c.r)
process_time = proc.time() - start_time
print("Generate data process time:")
```

```
print(process_time)

## End(Not run)
```

---

get_weight   *This function calculates the weight of each observation.*

Description

In the weighted least squares method, the weights $w_{(i)}$ are the jumps in Kaplan-Meier estimator based on $(Y_{(i)}, \delta_{(i)})$, i = 1,...,n.

$$w_{(1)} = \frac{\delta_{(1)}}{n},$$

$$w_{(i)} = \frac{\delta_{(i)}}{n-i+1} \cdot \prod_{j=1}^{i-1} \left(\frac{n-j}{n-j+1} \right)^{\delta_{(j)}}, i = 2, \ldots, n.$$

Usage

```
get_weight(x, y, status)
```

Arguments

x	The p-dimensional covariate matrix of n samples.
y	$Y_i = \min\{ln(T_i), ln(C_i)\}$, where C_i is the censoring time.
status	The censoring indicator.

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
data1=get_data(n, p, beta, varr1, alpha, mu2, varr2, c_r, seed)
x = data1[,:p]
y = data1[,p+2]
status = data1[,p+1]
```

```
start_time = proc.time()
weight = get_weight(x, y, status)
process_time = proc.time() - start_time
print("Generate data process time:")
print(process_time)

## End(Not run)
```

`get_weighted_data` *This function is used to generate and modify data with weights to fit standard least squares.*

Description

Let the design matrix be $X = (x_{(1)}, \ldots, x_{(n)})^T$ and let $Y = (Y_{(1)}, \ldots, Y_{(n)})^T$. Define

$$\tilde{X} = \text{diag}\left(\sqrt{w_{(1)}}, \ldots, \sqrt{w_{(n)}}\right) \cdot X,$$

$$\bar{Y} = \text{diag}\left(\sqrt{w_{(1)}}, \ldots, \sqrt{w_{(n)}}\right) \cdot Y.$$

Without loss of generality, assume that $\|\tilde{x}_j\|_2 > 0$, $j = 1, \ldots, p$, hold throughout this paper, where $\tilde{x}_j$ is the jth column of $\tilde{X}$. Let

$$D = \text{diag}\Big(\frac{\sqrt{n}}{\|\tilde{x}_1\|_2}, \ldots, \frac{\sqrt{n}}{\|\tilde{x}_p\|_2}\Big).$$

Define $\eta = D^{-1}\beta$ and $\bar{X} = \tilde{X}D$. Then each column of $\bar{X}$ is $\sqrt{n}$-length and supp(η)=supp(β), where supp(β)=$\{j : \beta_j \neq 0, j = 1, \ldots, p\}$. Let

$$L_2(\eta) = \frac{1}{2n}\left\|\bar{Y} - \bar{X}\eta\right\|_2^2.$$

Define

$$\eta^\diamond = \min_{\eta \in R^p} L_2(\eta) + \lambda\|\eta\|_0,$$

The estimator of β can be obtained as $\beta^\diamond = D\eta^\diamond$.

Usage

```
get_weighted_data(n, p, beta, varr1, alpha, mu2, varr2, c_r, seed = 1L)
```

Arguments

`n`	The sample size.
`p`	The variable dimension.
`beta`	The underlying regression coeffcient vector β
`varr1`	The standard error of normal distribution that generate $\widetilde{X}$. Default: `varr1=1`
`alpha`	A measure of the correlation among covariates
`mu2`	ϵ_i is generated independently from $N(mu2, varr2^2)$. Default: `mu2=0`
`varr2`	ϵ_i is generated independently from $N(mu2, varr2^2)$
`c_r`	The censoring rate
`seed`	Random seed. Default: `seed=1L`

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
start_time = proc.time()
data1=get_weighted_data(n,p,beta,varr1,alpha,mu2,varr2,c.r)
process_time = proc.time() - start_time
print("Generate data process time:")
print(process_time)

## End(Not run)
```

`Sdar` *The SDAR algorithm that is described in the paper 'A constructive approach to L0 penalized regression'*

Description

A constructive approach to estimating sparse, high-dimensional linear regression models. It generates a sequence of solutions iteratively, based on support detection using primal and dual information and root finding.

Usage

```
Sdar(ita0, T1, x, y, tau1, dd, iter_max)
```

Arguments

`ita0`	The initial input of η in SDAR algorithm.
`T1`	The initial input of failure time. Normally equals to the parameter tau in `Asder` function.
`x`	The motified X, which can be obtained by `get_weighted_data`.
`y`	The motified Y, which can be obtained by `get_weighted_data`.
`tau1`	The step size 0<τ<1 in the definitions of the active and inactive sets. Default: `tau1=1`
`dd`	The diagonal element vector of matrix D, $\frac{\sqrt{n}}{\|\tilde{x}_i\|_2}$, i=1,...,p.
`iter_max`	A maximum number of iterations.

Author(s)

Xingdong Feng, Jian Huang, Yuling Jiao, Shuang Zhang.

Maintainer: Shuang Zhang <zhangshuang_jz@sina.cn>

Examples

```
## Not run:
varr1=1
varr2=1
mu2=0
c.r=0.3
alpha=0.3
n=500
p=10000
m1=varr2*sqrt(2*log(p)/n)
m2=R*m1
set.seed(i)
b1=runif(T1,m1,m2)
beta=rep(0,p)
beta[supp.true[[i]]]=b1
data1=get_weighted_data(n,p,beta,varr1,alpha,mu2,varr2,c.r)
tau1=1
iter.max=20
ita0=rep(0,10000)
T1=20
x.ita=data1[[1]]
y.ita=data1[[2]]
d = data1[[4]]
res = Sdar(ita0,T1,x.ita,y.ita,tau1,d,iter.max)

## End(Not run)
```

Index