

Package ‘LassoNet’

January 19, 2020

Type Package

Title 3CoSE Algorithm

Version 0.8.3

Date 2019-12-17

Author Jonas Striaukas [aut, trl, cre] and Matthias Weber [aut]

Maintainer Jonas Striaukas <jonas.striaukas@gmail.com>

Description Contains functions to estimate a penalized regression model using 3CoSE algorithm, see Weber, Striaukas, Schumacher Binder (2018) <doi:10.2139/ssrn.3211163>.

License GPL (>= 2)

Imports Rcpp (>= 0.11.5)

Suggests snowfall

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-01-19 15:30:08 UTC

R topics documented:

LassoNet-package	2
beta.update.net	2
betanew_lasso_cpp	4
fastols	5
get.BxBy	6
get.signs.M	7
get.xi	7
lasso.net.fixed	8
lasso.net.grid	9
mat.to.laplacian	11
matrix.M.update	12
soft.thresh	13

Index	14
--------------	-----------

LassoNet-package *LassoNet: package for 3CoSE algorithm.*

Description

LassoNet contains functions to estimate a penalized regression model using 3CoSE algorithm described in the paper Weber, Striaukas, Schumacher and Binder (2018). The main function of the package is the function `lasso.net.grid`, see the example below.

Details

Package: LassoNet
Type: Package
Version: 0.8.3
Date: 2019-12-16
License: Open source

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

See Also

[Rcpp](#), [glmnet](#)

beta.update.net *Updates β coefficients.*

Description

This function updates β for given penalty parameters.

Usage

```
beta.update.net(x,y,beta,lambda1,lambda2,M1,n.iter,iscpp,tol)
```

Arguments

x	input data matrix of size $n \times p$; n - number of observations; p - number of covariates
y	response vector or size $n \times 1$
beta	initial value for β ; default - zero vector of size $n \times 1$
lambda1	lasso penalty parameter
lambda2	network penalty parameter
M1	penalty matrix
n.iter	maximum number of iterations for β step; default - 1e5
iscpp	binary choice for using cpp function in coordinate updates; 1 - use C++ (default), 0 - use R
tol	convergence tolerance level; default - 1e-6

Details

Updates the coefficient vector β given the data and penalty parameters λ_1 and λ_2 . Convergence criterion is defined as $\sum_{i=1}^p |\beta_{i,j} - \beta_{i,j-1}| \leq \text{to}$.

Value

beta	updated β vector
convergence	binary variable; 1 - yes
steps	number of steps until convergence

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

Examples

```
p<-200
n<-100
beta.0=array(1,c(p,1))
x<-matrix(rnorm(n*p),n,p)
y<-rnorm(n,mean=0,sd=1)
lambda1<-1
lambda2<-1
M1<-diag(p)
updates<-beta.update.net(x, y, beta.0, lambda1, lambda2, M1)
```

betanew_lasso_cpp *C++ subroutine that updates β coefficients.*

Description

This function updates β for given penalty parameters.

Usage

```
betanew_lasso_cpp(xx, xy, beta, M, y, Lambda1, Lambda2, iter, tol)
```

Arguments

xx	Bx matrix
xy	By vector
beta	initial value for β ; default - zero vector of size $p \times 1$
M	penalty matrix
y	response vector or size $n \times 1$
Lambda1	lasso penalty parameter
Lambda2	network penalty parameter
iter	maximum number of iterations for β step
tol	convergence tolerance level

Details

See beta.update.net

Value

beta	updated β vector
steps	number of steps until convergence

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <[doi:10.2139/ssrn.3211163](https://doi.org/10.2139/ssrn.3211163)>

Examples

```
p<-200
n<-100
beta.0=array(1,c(p,1))
x<-matrix(rnorm(n*p),n,p)
y<-rnorm(n,mean=0,sd=1)
lambda1<-1
lambda2<-1
M1<-diag(p)
updates<-beta.update.net(x, y, beta.0, lambda1, lambda2, M1)
```

fastols

Fast least squares estimate.

Description

Computes least squares estimate in an efficient way.

Usage

```
fastols(y, x)
```

Arguments

y	dependent variable
x	response variable

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

Examples

```
p<-10
n<-100
x<-matrix(rnorm(n*p),n,p)
beta<-array(5, c(p,1))
y<-x%*%beta + rnorm(n,mean=0,sd=0.1)
fastols(y,x)
```

get.BxBy *Computes decomposition elements.*

Description

Computes matrices B_X^{ij} and B_Y^{ij} to speed up estimation of connection signs. These matrices are stored only for indices that have non zero entries in penalty matrix M.

Usage

```
get.BxBy(x, y, M)
```

Arguments

x	Input data matrix of size $n \times p$, n - number of observations, p - number of covariates
y	y Response vector or size $n \times 1$
M	penalty matrix

Details

Calculates matrices all for i and j indices that have non zero values in a given penalty matrix.

Value

Bx	array of B_X^{ij} stored matrices. $Bx[, k]$ are the k-th combination of i and j non zero entry in the penalty matrix M
By	array of B_Y^{ij} stored matrices. $By[, k]$ are the k-th combination of i and j non zero entry in the penalty matrix M

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

Examples

```
p<-200
n<-100
x<-matrix(rnorm(n*p),n,p)
y<-rnorm(n,mean=0,sd=1)
M<-diag(p)
get.BxBy(x, y, M)
```

get.signs.M	<i>Vectorizes connection sign matrix.</i>
-------------	---

Description

Stores a matrix of connection signs to a vector.

Usage

```
get.signs.M(MAT)
```

Arguments

MAT	matrix of connection signs that contains -1, 1 or 0
-----	---

Value

vec.out	vectorized MAT matrix
---------	-----------------------

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

get.xi	<i>Updates the estimates of the connection signs by running mini OLS models.</i>
--------	--

Description

Updates connection signs $\hat{\xi}$.

Usage

```
get.xi(Bx,By,beta,xi,M)
```

Arguments

Bx	Bx element
By	By element
beta	$\hat{\beta}$ estimated value
xi	$\hat{\xi}$ matrix estimated at the previous step
M	penalty matrix

Value

xi	$\hat{\xi}$ matrix
----	--------------------

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

lasso.net.fixed *Estimates coefficients over the grid values of penalty parameters.*

Description

See lasso.net.grid

Usage

```
lasso.net.fixed(x,y,beta.0,lambda1,lambda2,M1,n.iter,iscpp,tol)
```

Arguments

x	$n \times p$ input data matrix
y	response vector or size $n \times 1$
beta.0	initial value for β ; default - zero vector of size $n \times 1$
lambda1	lasso penalty coefficient
lambda2	network penalty coefficient
M1	penalty matrix
n.iter	maximum number of iterations for β updating; default - 1e5
iscpp	binary choice for using cpp function in coordinate updates; 1 - use C++ (default), 0 - use R.
tol	convergence in β tolerance level; default - 1e-6

Details

Function loops through the grid of values of penalty parameters λ_1 and λ_2 until convergence is reached. Warm starts are stored for each iterator. The warm starts are stored once the coordinate updating converges.

Value

beta	Matrix of β coefficients. Columns denote different λ_1 coefficients, rows - λ_2 coefficients
mse	Mean squared error value
iterations	matrix with stored number of steps for sign matrix to converge
update.steps	matrix with stored number of steps for β updates to converge. (only stores the last values from connection signs iterations)
convergence.in.grid	matrix with stored values for convergence in β coefficients. If at least one β did not converge in sign matrix iterations, 0 (false) is stored, otherwise 1 (true)

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

Examples

```
p=200
n=100
beta.0=array(1,c(p,1))
x=matrix(rnorm(n*p),n,p)
y=rnorm(n,mean=0,sd=1)
lambda1=c(0,1)
lambda2=c(0,1)
M1=diag(p)
lasso.net.fixed(x, y, beta.0, lambda1, lambda2, M1)
```

lasso.net.grid	<i>Estimates coefficients and connection signs over the grid of values of penalty parameters λ_1 and λ_2.</i>
----------------	---

Description

Fits network regressions over the grid of values of penalty parameters λ_1 and λ_2 , stores connection signs, number of iterations until convergence and convergence outcome.

Usage

```
lasso.net.grid(x,y ,beta.0,lambda1,lambda2,M1,m.iter,n.iter,iscpp=TRUE,tol,alt.num)
```

Arguments

<code>x</code>	$n \times p$ input data matrix
<code>y</code>	response vector or size $n \times 1$
<code>beta.0</code>	initial value for β . default - zero vector of size $n \times 1$
<code>lambda1</code>	lasso penalty coefficient
<code>lambda2</code>	network penalty coefficient
<code>M1</code>	penalty matrix
<code>m.iter</code>	maximum number of iterations for sign matrix updating; default - 100
<code>n.iter</code>	maximum number of iterations for β updating; default - 1e5
<code>iscpp</code>	binary choice for using cpp function in coordinate updates; 1 - use C++ (default), 0 - use R
<code>tol</code>	convergence in β tolerance level; default - 1e-6
<code>alt.num</code>	alt.num remaining iterations are stored; default - 12

Details

Fits network regression for the grid values of λ_1 and λ_2 using warm starts.

Value

<code>beta</code>	matrix of β coefficients, columns are for different λ_1 parameters, rows λ_2 parameters
<code>mse</code>	mean squared error value
<code>M</code>	array of connection signs. $M[, i, j]$ is the connection sign matrix for j-th λ_1 value and i-th λ_2 value
<code>iterations</code>	matrix with stored number of steps for sign matrix to converge
<code>update.steps</code>	matrix with stored number of steps for β updates to converge. (only stores the last values from connection signs iterations)
<code>convergence.in.M</code>	matrix with stored values for convergence in sign matrix
<code>convergence.in.grid</code>	matrix with stored values for convergence in β coefficients. If at least one β did not converge in sign matrix iterations, 0 (false) is stored, otherwise 1 (true)
<code>xi.conv</code>	array with stored connection signs changes in each iteration
<code>beta.alt</code>	array of coefficient vectors in case connection signs alternate

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

Examples

```
p=200
n=100
beta.0=array(1,c(p,1))
x=matrix(rnorm(n*p),n,p)
y=rnorm(n,mean=0,sd=1)
lambda1=c(0,1)
lambda2=c(0,1)
M1=diag(p)
lasso.net.grid(x, y, beta.0, lambda1, lambda2, M1)
```

mat.to.laplacian	<i>Computes Laplacian matrix.</i>
------------------	-----------------------------------

Description

Computes Laplacian matrix.

Usage

```
mat.to.laplacian(M1, type)
```

Arguments

M1	$p \times p$ matrix
type	Laplacian types: 1) "normalized" (default) - normalized Laplacian, 2) "combinatorial" - combinatorial Laplacian

Value

L	Laplacian
---	-----------

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

matrix.M.update *Updates connection sign matrix.*

Description

Updates M using relation $(M)_{ij} = -\hat{\xi}_{ij}|(M_1)_{ij}$.

Usage

```
matrix.M.update(M, xi)
```

Arguments

M	penalty matrix
xi	estimated $\hat{\xi}_{ij}$ matrix

Details

Updates M

Value

M	updated M
---	-----------

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

References

Weber, M., Striaukas, J., Schumacher, M., Binder, H. "Network-Constrained Covariate Coefficient and Connection Sign Estimation" (2018) <doi:10.2139/ssrn.3211163>

Examples

```
p<-100
M<-diag(p)
xi<-matrix(rnorm(p*p), p, p)
matrix.M.update(M,xi)
```

soft.thresh	<i>Soft thresholding operator.</i>
-------------	------------------------------------

Description

Soft thresholding operator.

Usage

```
soft.thresh(x, kappa)
```

Arguments

x	β coordinate
kappa	κ value in general or λ_1 for covariance updating

Details

Soft thresholding definition: $S(x, \kappa) = \text{sign}(x)(|x| - \kappa)_+$

Value

x	value after applying soft thresholding operator
---	---

Author(s)

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

Examples

```
kappa<-0.2  
x<-0.7  
soft.thresh(x, kappa)
```

Index

`beta.update.net`, [2](#)
`betanew_lasso_cpp`, [4](#)

`fastols`, [5](#)

`get.BxBy`, [6](#)
`get.signs.M`, [7](#)
`get.xi`, [7](#)
`glmnet`, [2](#)

`lasso.net.fixed`, [8](#)
`lasso.net.grid`, [9](#)
`LassoNet-package`, [2](#)

`mat.to.laplacian`, [11](#)
`matrix.M.update`, [12](#)

`Rcpp`, [2](#)

`soft.thresh`, [13](#)