Package 'LassoNet'

April 28, 2018

Type Package

Title LassoNet: package for 3CoSE algorithm

Version 0.8.3

Date 2018-04-27

Author Jonas Striaukas

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

Description LassoNet package contains functions to estimate a penalized regression model using 3CoSE algorithm described in the paper Striaukas and Weber (2018). Regression model has two penalty terms: 1) standard lasso and 2) network. Using this package, you can estimate both regression coefficients and network connection signs. In addition, there is a separate file (replicated.R) that replicates simulation results of the paper Striaukas and Weber (2018).

License GPL (>= 2)

Imports Rcpp (>= 0.11.5)

Suggests snowfall

LinkingTo Rcpp

NeedsCompilation yes

R topics documented:

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

Description

LassoNet R package estimates the penalized regression model slope coefficients together with connection signs between covariates using the 3CoSE algorithm described in the paper Striaukas and Weber (2018). The regression coefficients are penalized both in ℓ_1 and ℓ_2 norms, where the latter uses a network penalty matrix. The algorithm uses warm starts for the β vector and connection sign matrix M. The main function of the package is lasso.net.grid, see example below.

Details

Package:	LassoNet
Type:	Package
Version:	0.8.3
Date:	2018-04-27
License:	Open source

Author(s)

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

References

Weber, M., Schumacher, M., & Binder, H. (2014) *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018) *Network Constrained Covariate Coefficient and Connection Sign Estimation*

See Also

`Rcpp`, `glmnet`

`beta.update.net` *Updates β coefficients.*

Description

This function updates β coefficients for a 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 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 tolerance level. default - 1e-6

Details

The function updates the coefficient vector β given the data and penalty parameters. Convergence criterion is defined as: $\sum_{i=1}^{p} |\beta_{i,j} - \beta_{i,j-1}| \leq$ tol.

Value

`beta`	updated β vector
`convergence`	binary variable for convergence
`steps`	number of steps until convergence

Author(s)

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

References

Weber, M., Schumacher, M., & Binder, H. (2014) *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018) *Network Constrained Covariate Coefficient and Connection Sign Estimation*

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 coefficients for a 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 $n \times 1$
M	penalty matrix
y	response vector or size $n \times 1$
Lambda1	lasso penalty coefficient
Lambda2	network penalty coefficient
iter	maximum number of iterations for β updating.
tol	convergence tolerance level.

Details

The function updates the coefficient vector β given the data and penalty parameters. Convergence criterion is defined as: $\sum_{i=1}^{p} |\beta_{i,j} - \beta_{i,j-1}| \leq$ tol.

Value

beta	updated β vector
steps	number of steps until convergence

Author(s)

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

References

Weber, M., Schumacher, M., & Binder, H. (2014) *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018) *Network Constrained Covariate Coefficient and Connection Sign Estimation*

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)
```

drawdata *Draw random data.*

Description

Function generates random data.

Usage

```
drawdata(n, beta)
```

Arguments

n	number of observations
beta	β coefficient vector

Author(s)

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

Examples

```
n<-10
beta<-rep(0.2,n)
data<-drawdata(n, beta)
```

fastols *Fast least squares estimate.*

Description

Function 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
fastols(y,x)
```

---

get.BxBy *Computes decomposition elements, p.g. 14 of Weber et. al (2014).*

Description

Function computes matrices B_X^{ij} and B_y^{ij} to speed up estimation of connection signs. These matrices are precomputed, stored. We store only 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

Function calculates matrices all for i and j indices that have non zero values in a given penalty matrix, since ξ matrix is updated only for such indices.

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., Schumacher, M., & Binder, H. (2014) *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018) *Network Constrained Covariate Coefficient and Connection Sign Estimation*

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` *A function that vetorizes connetion sign matrix.*

Description

Stores a matrix of connection signs into a vector.

Usage

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

Arguments

`MAT`	A 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` *Updates the estimates of the connection signs by running mini OLS models as described in Striaukas and Weber (2018).*

Description

Function 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`	Updated $\hat{\xi}$ matrix

Author(s)

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

References

Weber, M., Schumacher, M., & Binder, H. (2014). *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018). *Network Constrained Covariate Coefficient and Connection Sign Estimation*

`lasso.net.fixed` *Estimates coefficients and connection signs over the grid values (λ1, λ2) where connection signs are kept unchanged.*

Description

The function is largely the same as lasso.net.grid, but the network connection signs are kept unchanged.

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 grid values of $\lambda1$ and $\lambda2$ until convergence. Warm starts are stored for each iterator. The warm starts are stored once coordinate updates converge.

Value

`beta` Matrix of β coefficients. Columns denote different $\lambda1$ coefficients, rows - $\lambda2$ 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., Schumacher, M., & Binder, H. (2014). *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M (2018). *Network Constrained Covariate Coefficient and Connection Sign Estimation*

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)
```

`lasso.net.grid` *Estimates coefficients and connection signs over the grid values ($\lambda1$, $\lambda2$).*

Description

Function fits lasso and network regressions over the grid values of penalty coefficients. β updates are computed in R and C++. In addtion, function stores connection signs, number of iterations until convergence, convergence binary output and alternating signs in case divergence in connection matrix.

Usage

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

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
`m.iter`	maximum number of iterations for sign matrix updating. default - 100
`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
`alt.num`	In case connection sign matrix alternates, alt.num last iterataions are stored. default - 12

Details

Function loops through (using two for loops and one while loop) grid values of $\lambda 1$ and $\lambda 2$ until either sign matrix convergences or maximum (m.iter) number of iterations is reached. Convergence in sign matrix is checked by evaluating whether two consecutive matrices are equal. Warm starts are stored for each iterator. The first warm start is stored once coordinate updates converge given initial value for sign matrix (i.e. first iteration in sign matrix). Warm start for a new value of $\lambda 2$ is stored when coordinate updates converge given initial value for sign matrix (i.e. first iteration in sign matrix, but last iteration in $\lambda 1$). In addition, function checks whether signs alternate. Once the algorithm is close to the maximum number of iterations (m.iter) in sign matrix, function stores two previous sign matrices and checks whether they alternate (wheter sign matrix equals to the previous two steps estimated sign matrix and does not equal to one step before estimated sign matrix). Further convergence checks are done for β values when looping though sign matrix. If at least one β updating procedure did not converge, `false` is stored in a convergece.in.grid output matrix while otherwise `true` is stored.

Value

`beta`	matrix of β coefficients. Columns denote different $\lambda 1$ coefficients, rows - $\lambda 2$ coefficients
`mse`	Mean squared error value
`M`	Array of connection signs. $M[,,i,j]$ is the connection sign matrix for j-th $\lambda 1$ value and i-th $\lambda 2$ 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.M`	matrix with stored values for convergence in sign matrix

`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)

`xi.conv` array with stored values about changes in connection signs (each iteration)

`beta.alt` vector for β value in the case when M1 matrix alternates, zero otherwise

Author(s)

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

References

Weber, M., Schumacher, M., & Binder, H. (2014) *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018) *Network Constrained Covariate Coefficient and Connection Sign Estimation*

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` *Computes laplacian given some matrix M.*

Description

Computes laplacian given some matrix M.

Usage

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

Arguments

`M1` $p \times p$ matrix

`type` Laplacian types. "normalized" (default) gives normalized laplacian, "combinatorial" gives combinatorial.

Value

`L` Laplacian

Author(s)

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

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

Description

Function updates M matrix 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

Updating equation is at Weber et. al (2014), p.g. 9, 2nd step of algorithm

Value

`M`	updated correct M matrix

Author(s)

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

References

Weber, M., Schumacher, M., & Binder, H. (2014) *Regularized Regression Incorporating Network Information: Simultaneous Estimation of Covariate Coefficients and Connection Signs*

Striaukas, J. & Weber, M. (2018) *Network Constrained Covariate Coefficient and Connection Sign Estimation*

Examples

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

soft.thresh *Soft thresholding operator.*

Description

Evaluates each element of β using soft thresholding operator.

Usage

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

Arguments

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

Details

Uses standard soft thresholding definition: $S(x, \kappa) = sign(x)(|x| - \kappa)_+$

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