Package 'CSGL'

June 5, 2018

Title Compositional Sparse Group Lasso

Version 0.0.0.9000

Description Fits a linear log-contrast model with L1 and L2 penalties using an alternating direction method of multipliers algorithm.

Depends R (>= 3.4.3)

License GPL

Encoding UTF-8

LazyData true

Imports SGL

RoxygenNote 6.0.1

NeedsCompilation no

Author Anna Plantinga [aut, cre]

Maintainer Anna Plantinga <amp9@williams.edu>

R topics documented:

calc.gic *Calculates generalized information criterion*

Description

Calculates GIC as defined in Lin (2014).

Usage

```
calc.gic(Z, y, betahat, int = 0)
```

Arguments

Z	Matrix of log OTU proportions
y	Outcome vector
betahat	Estimated coefficient vector
int	Estimated intercept; default = 0

Value

GIC

calc.lammax *Calculates maximum lambda value*

Description

Calculates maximum lambda value as smallest lambda for which all coefficients are zero.

Usage

```
calc.lammax(Z, y, n, p, H, mu, theta, groups, thresh, erel, maxit)
```

Arguments

Z	Matrix of log OTU proportions
y	Outcome vector
n	Number of subjects
p	Number of OTUs
H	Matrix I_p - (1/p) 1_p 1_p'
mu	Augmented Lagrangian parameter
theta	Defines convex combination between L1 and L2 penalties (theta = 1 is L1 only, theta = 0 is L2 only)
groups	Vector indicating group membership of each OTU
thresh	Threshold for convergence
erel	Internal parameter for ADMM convergence
maxit	Maximum number of iterations

Value

Maximum lambda value

`csgl` *Compositional sparse group lasso*

Description

Fits CSGL for a sequence of lambda values.

Usage

```
csgl(Z, y, groups, mu = 1, theta = 0.95, lam.seq = NULL, nlam = 25,
  min.frac = 0.1, thresh = 1e-07, maxit = 10000, std = T, verbose = F)
```

Arguments

`Z`	Matrix of log OTU proportions
`y`	Outcome vector
`groups`	Vector indicating group membership of each OTU
`mu`	Augmented Lagrangian parameter
`theta`	Defines convex combination between L1 and L2 penalties (theta = 1 is L1 only, theta = 0 is L2 only)
`lam.seq`	Sequence of lambda values to consider
`nlam`	Number of lambda values to try
`min.frac`	Minimum value of the penalty parameter, as a fraction of the maximum value
`thresh`	Threshold for convergence
`maxit`	Maximum number of iterations
`std`	Logical flag for variable standardization before fitting the model
`verbose`	Logical flag for printing updates during model fitting

Value

List with components

`int`	Vector of intercepts, one per lambda
`beta`	Matrix of coefficients, one column per lambda
`lambdas`	Sequence of lambda values

csgl.fit *Fits CSGL*

Description

Function to fit CSGL. Interior function for csgl (sequence of lambdas), gic.csgl (choosing best lambda by GIC), and cv.csgl (choosing best lambda by k-fold cross-validation).

Usage

```
csgl.fit(Z, y, n, p, H, mu, lambda, theta, groups, thresh, erel, maxit)
```

Arguments

Z	Matrix of log OTU proportions
y	Outcome vector
n	Number of subjects
p	Number of OTUs
H	Matrix I_p - (1/p) 1_p 1_p'
mu	Augmented Lagrangian parameter
lambda	Regularization parameter
theta	Defines convex combination between L1 and L2 penalties (theta = 1 is L1 only, theta = 0 is L2 only)
groups	Vector indicating group membership of each OTU
thresh	Threshold for convergence
erel	Internal parameter for ADMM convergence
maxit	Maximum number of iterations

Value

Fitted coefficients beta

cv.csgl *Fits CSGL, choosing lambda by cross-validation*

Description

Fits CSGL for a sequence of lambda values and chooses lambda based on k-fold cross-validation.

Usage

```
cv.csgl(Z, y, groups, mu = 1, theta = 0.95, lam.seq = NULL, nlam = 25,
  min.frac = 0.1, nfolds = 10, thresh = 1e-07, maxit = 10000, std = T,
  verbose = F)
```

Arguments

Z	Matrix of log OTU proportions
y	Outcome vector
groups	Vector indicating group membership of each OTU
mu	Augmented Lagrangian parameter
theta	Defines convex combination between L1 and L2 penalties (theta = 1 is L1 only, theta = 0 is L2 only)
lam.seq	Sequence of lambda values to consider
nlam	Number of lambda values to try
min.frac	Minimum value of the penalty parameter, as a fraction of the maximum value
nfolds	Number of folds for cross-validation
thresh	Threshold for convergence
maxit	Maximum number of iterations
std	Logical flag for variable standardization before fitting the model
verbose	Logical flag for printing updates during model fitting

Value

A list with components

cv.int	Intercept for lambda chosen by CV
cv.beta	Coefficient vector for lambda chosen by CV
cv.lam	Lambda chosen by CV
lamseq	Sequence of lambda values tried
pe	Cross-validated prediction error for each lambda

---

getFolds *Gets folds for cross-validation*

Description

Gets folds for cross-validation

Usage

```
getFolds(n, nfolds)
```

Arguments

n	Number of subjects
nfolds	Desired number of folds

Value

List of folds

`gic.csgl` *Fits CSGL, choosing lambda by GIC*

Description

Fits CSGL for a sequence of lambdas and chooses lambda based on the GIC.

Usage

```
gic.csgl(Z, y, groups, mu = 1, theta = 0.95, lam.seq = NULL, nlam = 25,
  min.frac = 0.1, thresh = 1e-07, maxit = 10000, std = T, verbose = F)
```

Arguments

`Z`	Matrix of log OTU proportions
`y`	Outcome vector
`groups`	Vector indicating group membership of each OTU
`mu`	Augmented Lagrangian parameter
`theta`	Defines convex combination between L1 and L2 penalties (theta = 1 is L1 only, theta = 0 is L2 only)
`lam.seq`	Sequence of lambda values to consider
`nlam`	Number of lambda values to try
`min.frac`	Minimum value of the penalty parameter, as a fraction of the maximum value
`thresh`	Threshold for convergence
`maxit`	Maximum number of iterations
`std`	Logical flag for variable standardization before fitting the model
`verbose`	Logical flag for printing updates during model fitting

Value

A list with components

`gic.int`	Intercept for lambda chosen by GIC
`gic.beta`	Coefficient vector for lambda chosen by GIC
`fullfit`	Full model fit (all lambda values)
`gic`	Vector of GIC values for each lambda

update.alpha *Update alpha vector*

Description

This function updates alpha by projecting (beta + xi) onto the subspace defined by sum(alpha) = 0.

Usage

```
## S3 method for class 'alpha'
update(H, beta, xi, p)
```

Arguments

H	Matrix I_p - (1/p) 1_p 1_p'
beta	p-vector of coefficients
xi	p-vector of Lagrange multipliers
p	Number of OTUs

Value

Updated alpha vector

update.beta *Update beta vector*

Description

This function uses the SGL package to minimize the augmented Lagrangian with respect to beta.

Usage

```
## S3 method for class 'beta'
update(y, Z, n, p, mu, alpha, xi, lambda, theta, groups)
```

Arguments

y	Outcome vector
Z	Matrix of log OTU proportions
n	Number of subjects (nrow(Z))
p	Number of OTUs (ncol(Z))
mu	Augmented Lagrangian parameter
alpha	p-vector of coefficients
xi	p-vector of Lagrange multipliers
lambda	Regularization parameter
theta	Defines convex combination between L1 and L2 penalties (theta = 1 is L1 only)
groups	Vector indicating group membership of each OTU

Value

Updated beta vector

`update.xi` *Update xi vector*

Description

This function updates the Lagrange multipliers xi.

Usage

```
## S3 method for class 'xi'
update(alpha, beta, xi)
```

Arguments

`alpha`	p-vector of coefficients
`beta`	p-vector of coefficients
`xi`	p-vector of Lagrange multipliers

Value

Updated xi vector

Index