Package 'lsgl'

September 20, 2015

Type Package

Title Linear Sparse Group Lasso

Version 1.2.0

Date 2015-09-16

Author Martin Vincent

Maintainer Martin Vincent <vincent@math.ku.dk>

Description Linear multiple output using sparse group lasso. The algorithm finds the sparse group lasso penalized maximum likelihood estimator. This result in feature and parameter selection, and parameter estimation. Use of multiple processors for cross validation and subsampling is supported through OpenMP. Development version is on github.

URL https://github.com/vincent-dk/lsgl

License GPL (>= 2)

LazyLoad yes

Imports methods, utils, stats

Depends R (>= 3.0.0), sglOptim (== 1.2.0), Matrix

LinkingTo sglOptim, Rcpp, RcppProgress, RcppArmadillo, BH

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-09-19 22:47:05

R topics documented:

`coef.lsgl` *Extract nonzero coefficients*

Description

Extract nonzero coefficients

Usage

```
## S3 method for class 'lsgl'
coef(object, index = 1:nmod(object), ...)
```

Arguments

`object`	a lsgl object
`index`	indices of the models
`...`	ignored

Value

a list of length `length(index)` with nonzero coefficients of the models

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of covariates
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)
```

```
lambda<-lsgl.lambda(X,Y, alpha=1, d = 25, lambda.min=.5, intercept=FALSE)
fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE)

# the nonzero coefficients of the models 1, 2 and 5
coef(fit, index = c(1,2,5))
```

`Err.lsgl` *Compute error rates*

Description

Compute and return an error for each model. The error may be spicifed in the `loss` argument.

The root-mean-square error (RMSE) is

$$\frac{1}{K}\sum_{i=1}^{K}\sqrt{\frac{1}{N}\sum_{j=1}^{N}Y_{ji}-(X\hat{\beta})_{ji}}$$

RMSE is the default error.

The objective value error (OVE) is

$$\|Y - X\hat{\beta}\|_F$$

The scaled objective value error (SOVE) is

$$\frac{1}{NK}\|Y - X\hat{\beta}\|_F$$

Usage

```
## S3 method for class 'lsgl'
Err(object, data = NULL, response = object$Y.true,
  y = response, loss = "RMSE", ...)
```

Arguments

`object`	a lsgl object.
`data`	a design matrix (the X matrix).
`response`	redirected to `y`.
`y`	a matrix of the true responses (the Y matrix).
`loss`	the loss (error) function. Either a function taking two arguments or one of the following character strings `RMSE`, `OVE` or `SOVE`.
`...`	ignored.

Value

a vector of errors.

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of features
K <- 15  #number of groups

# simulate beta matrix and X matrix
B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)
X1<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y1 <-X1%*%B+matrix(rnorm(N*K,0,1),N,K)

X2<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y2 <-X2%*%B+matrix(rnorm(N*K,0,1),N,K)

#### Fit models using X1
lambda <- lsgl.lambda(X1, Y1, alpha = 1, d = 25L, lambda.min = 5, intercept = FALSE)
fit <- lsgl(X1, Y1, alpha = 1, lambda = lambda, intercept = FALSE)

## Training errors:
Err(fit, X1)

## Errors predicting Y2:
Err(fit, X2, Y2)

#### Do cross validation
fit.cv <- lsgl.cv(X1, Y1, alpha = 1, lambda = lambda, intercept = FALSE)

## Cross validation errors (estimated expected generalization error)
Err(fit.cv)

## Cross validation errors using objective value error measures
Err(fit.cv, loss = "OVE")
```

---

features.lsgl *Nonzero features*

Description

Extracts the nonzero features for each model.

Usage

```
## S3 method for class 'lsgl'
features(object, ...)
```

Arguments

object	a lsgl object
...	ignored

Value

a list of of length nmod(x) containing the nonzero features (that is nonzero columns of the beta matrices)

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of covariates
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)

lambda<-lsgl.lambda(X,Y, alpha=1, d = 25, lambda.min=.5, intercept=FALSE)
fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE)

# the nonzero features of model 1, 10 and 25
features(fit)[c(1,10,25)]

# count the number of nonzero features in each model
sapply(features(fit), length)
```

---

lsgl *Fit a linear multiple output model using sparse group lasso*

Description

For a linear multiple output model with p features (covariates) dived into m groups using sparse group lasso.

Usage

```
lsgl(x, y, intercept = TRUE, weights = NULL, grouping = factor(1:ncol(x)),
  groupWeights = c(sqrt(ncol(y) * table(grouping))),
  parameterWeights = matrix(1, nrow = ncol(y), ncol = ncol(x)), alpha = 1,
  lambda, algorithm.config = lsgl.standard.config)
```

Arguments

`x`	design matrix, matrix of size $N \times p$.
`y`	response matrix, matrix of size $N \times K$.
`intercept`	should the model include intercept parameters.
`weights`	sample weights, vector of size $N \times K$.
`grouping`	grouping of features, a factor or vector of length p. Each element of the factor/vector specifying the group of the feature.
`groupWeights`	the group weights, a vector of length m (the number of groups).
`parameterWeights`	a matrix of size $K \times p$.
`alpha`	the α value 0 for group lasso, 1 for lasso, between 0 and 1 gives a sparse group lasso penalty.
`lambda`	lambda sequence.
`algorithm.config`	the algorithm configuration to be used.

Details

This function computes a sequence of minimizers (one for each lambda given in the `lambda` argument) of

$$\frac{1}{N}\|Y - X\beta\|_F^2 + \lambda\left((1-\alpha)\sum_{J=1}^{m}\gamma_J\|\beta^{(J)}\|_2 + \alpha\sum_{i=1}^{n}\xi_i|\beta_i|\right)$$

where $\|\cdot\|_F$ is the frobenius norm. The vector $\beta^{(J)}$ denotes the parameters associated with the J'th group of features. The group weights are denoted by $\gamma \in [0,\infty)^m$ and the parameter weights by $\xi \in [0,\infty)^n$.

Value

`beta`	the fitted parameters – the list $\hat{\beta}(\lambda(1)), \ldots, \hat{\beta}(\lambda(d))$ of length `length(return)`. With each entry of list holding the fitted parameters, that is matrices of size $K \times p$ (if `intercept = TRUE` matrices of size $K \times (p+1)$)
`loss`	the values of the loss function.
`objective`	the values of the objective function (i.e. loss + penalty).
`lambda`	the lambda values used.

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 50 #number of samples
p <- 50 #number of features
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)

lambda<-lsgl.lambda(X,Y, alpha=1, lambda.min=.5, intercept=FALSE)

fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE)

## ||B - \beta||_F
sapply(fit$beta, function(beta) sum((B - beta)^2))

## Plot
par(mfrow = c(3,1))
image(B, main = "True B")
image(as.matrix(fit$beta[[100]]), main = "Lasso estimate (lambda = 0.5)")
image(solve(t(X)%*%X)%*%t(X)%*%Y, main = "Least squares estimate")

# The training error of the models
Err(fit, X)
# In this cases this is simply the loss function
1/(sqrt(N)*K)*sqrt(fit$loss)
```

`lsgl.algorithm.config` *Create a new algorithm configuration*

Description

With the exception of `verbose` it is not recommended to change any of the default values.

Usage

```
lsgl.algorithm.config(tolerance_penalized_main_equation_loop = 1e-10,
  tolerance_penalized_inner_loop_alpha = 1e-04,
  tolerance_penalized_inner_loop_beta = 1,
  tolerance_penalized_middel_loop_alpha = 0.01,
  tolerance_penalized_outer_loop_alpha = 0.01,
  tolerance_penalized_outer_loop_beta = 0,
  tolerance_penalized_outer_loop_gamma = 1e-05,
  use_bound_optimization = FALSE,
```

```
  use_stepsize_optimization_in_penalizeed_loop = TRUE,
  stepsize_opt_penalized_initial_t = 1, stepsize_opt_penalized_a = 0.1,
  stepsize_opt_penalized_b = 0.1, inner_loop_convergence_limit = 1e+05,
  verbose = TRUE)
```

Arguments

`tolerance_penalized_main_equation_loop`
: tolerance threshold.

`tolerance_penalized_inner_loop_alpha`
: tolerance threshold.

`tolerance_penalized_inner_loop_beta`
: tolerance threshold.

`tolerance_penalized_middel_loop_alpha`
: tolerance threshold.

`tolerance_penalized_outer_loop_alpha`
: tolerance threshold.

`tolerance_penalized_outer_loop_beta`
: tolerance threshold.

`tolerance_penalized_outer_loop_gamma`
: tolerance threshold.

`use_bound_optimization`
: if `TRUE` hessian bound check will be used.

`use_stepsize_optimization_in_penalizeed_loop`
: if `TRUE` step-size optimization will be used.

`stepsize_opt_penalized_initial_t`
: initial step-size.

`stepsize_opt_penalized_a`
: step-size optimization parameter.

`stepsize_opt_penalized_b`
: step-size optimization parameter.

`inner_loop_convergence_limit`
: inner loop convergence limit.

`verbose`
: If `TRUE` some information, regarding the status of the algorithm, will be printed in the R terminal.

Value

A configuration.

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of features
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)

# Create configuration
config <- lsgl.algorithm.config(verbose = FALSE)

lambda<-lsgl.lambda(X,Y, alpha=1, lambda.min=.5, intercept=FALSE, algorithm.config = config)

fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE, algorithm.config = config)
```

`lsgl.c.config` *Featch information about the C side configuration of the package*

Description

Featch information about the C side configuration of the package

Usage

```
lsgl.c.config()
```

Value

list

Author(s)

Martin Vicnet

lsgl.cv *Linear multiple output cross validation using multiple possessors*

Description

Linear multiple output cross validation using multiple possessors

Usage

```
lsgl.cv(x, y, intercept = TRUE, weights = NULL,
  grouping = factor(1:ncol(x)), groupWeights = c(sqrt(ncol(y) *
  table(grouping))), parameterWeights = matrix(1, nrow = ncol(y), ncol =
  ncol(x)), alpha = 1, lambda, fold = 10L, cv.indices = list(),
  max.threads = 2L, algorithm.config = lsgl.standard.config)
```

Arguments

x	design matrix, matrix of size $N \times p$.
y	response matrix, matrix of size $N \times K$.
intercept	should the model include intercept parameters.
weights	sample weights, vector of size $N \times K$.
grouping	grouping of features, a factor or vector of length p. Each element of the factor/vector specifying the group of the feature.
groupWeights	the group weights, a vector of length m (the number of groups).
parameterWeights	a matrix of size $K \times p$.
alpha	the α value 0 for group lasso, 1 for lasso, between 0 and 1 gives a sparse group lasso penalty.
lambda	the lambda sequence for the regularization path.
fold	the fold of the cross validation, an integer larger than 1 and less than $N + 1$. Ignored if `cv.indices != NULL`.
cv.indices	a list of indices of a cross validation splitting. If `cv.indices = NULL` then a random splitting will be generated using the `fold` argument.
max.threads	the maximal number of threads to be used.
algorithm.config	the algorithm configuration to be used.

Value

Yhat	the cross validation estimated response matrix
Y.true	the true response matrix, this is equal to the argument `y`
cv.indices	the cross validation splitting used
features	number of features used in the models
parameters	number of parameters used in the models.

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of features
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)
X1<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y1 <-X1%*%B+matrix(rnorm(N*K,0,1),N,K)

##Do cross validation
lambda <- lsgl.lambda(X1, Y1, alpha = 1, d = 15L, lambda.min = 5, intercept = FALSE)
fit.cv <- lsgl.cv(X1, Y1, alpha = 1, lambda = lambda, intercept = FALSE)

## Cross validation errors (estimated expected generalization error)
Err(fit.cv)
```

---

lsgl.lambda *Compute a lambda sequence for the regularization path*

Description

Computes a decreasing lambda sequence of length d. The sequence ranges from a data determined maximal lambda $\lambda_{\max}$ to the user inputed `lambda.min`.

Usage

```
lsgl.lambda(x, y, intercept = TRUE, weights = NULL,
  grouping = factor(1:ncol(x)), groupWeights = c(sqrt(ncol(y) *
  table(grouping))), parameterWeights = matrix(1, nrow = ncol(y), ncol =
  ncol(x)), alpha = 1, d = 100L, lambda.min,
  algorithm.config = lsgl.standard.config)
```

Arguments

x	design matrix, matrix of size $N \times p$.
y	response matrix, matrix of size $N \times K$.
intercept	should the model include intercept parameters.
weights	sample weights, vector of size $N \times K$.

`grouping`	grouping of features, a factor or vector of length p. Each element of the factor/vector specifying the group of the feature.
`groupWeights`	the group weights, a vector of length m (the number of groups).
`parameterWeights`	a matrix of size $K \times p$.
`alpha`	the α value 0 for group lasso, 1 for lasso, between 0 and 1 gives a sparse group lasso penalty.
`d`	the length of lambda sequence
`lambda.min`	the smallest lambda value in the computed sequence.
`algorithm.config`	the algorithm configuration to be used.

Value

a vector of length `d` containing the compute lambda sequence.

Author(s)

Martin Vincent

`lsgl.standard.config` *Standard algorithm configuration*

Description

```
lsgl.standard.config <- lsgl.algorithm.config()
```

Usage

```
lsgl.standard.config
```

Format

```
List of 14
 $ tolerance_penalized_main_equation_loop      : num 1e-10
 $ tolerance_penalized_inner_loop_alpha        : num 1e-04
 $ tolerance_penalized_inner_loop_beta         : num 1
 $ tolerance_penalized_middel_loop_alpha       : num 0.01
 $ tolerance_penalized_outer_loop_alpha        : num 0.01
 $ tolerance_penalized_outer_loop_beta         : num 0
 $ tolerance_penalized_outer_loop_gamma        : num 1e-05
 $ use_bound_optimization                      : logi FALSE
 $ use_stepsize_optimization_in_penalizeed_loop: logi TRUE
 $ stepsize_opt_penalized_initial_t            : num 1
 $ stepsize_opt_penalized_a                    : num 0.1
 $ stepsize_opt_penalized_b                    : num 0.1
 $ inner_loop_convergence_limit                : int 100000
 $ verbose                                     : logi TRUE
```

Author(s)

Martin Vicnet

`models.lsgl` *Exstract the fitted models*

Description

Returns the fitted models, that is the estimated β matrices.

Usage

```
## S3 method for class 'lsgl'
models(object, index = 1:nmod(object), ...)
```

Arguments

`object`	a lsgl object
`index`	indices of the models to be returned
`...`	ignored

Value

a list of β matrices.

Author(s)

Martin Vincent

`nmod.lsgl` *Returns the number of models in a lsgl object*

Description

Returns the number of models in a lsgl object

Usage

```
## S3 method for class 'lsgl'
nmod(object, ...)
```

Arguments

`object`	a lsgl object
`...`	ignored

Value

the number of models in `object`

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of features
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)

lambda<-lsgl.lambda(X,Y, alpha=1, d = 25, lambda.min=.5, intercept=FALSE)
fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE)

# the number of models
nmod(fit)
```

---

parameters.lsgl *Nonzero parameters*

Description

Extracts the nonzero parameters for each model.

Usage

```
## S3 method for class 'lsgl'
parameters(object, ...)
```

Arguments

`object`	a lsgl object
`...`	ignored

Value

a list of length `nmod(x)` containing the nonzero parameters of the models.

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of features
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)

lambda<-lsgl.lambda(X,Y, alpha=1, d = 25, lambda.min=.5, intercept=FALSE)
fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE)

# the nonzero parameters of model 1, 10 and 25
parameters(fit)[c(1,10,25)]

# count the number of nonzero parameters in each model
sapply(parameters(fit), sum)
```

---

predict.lsgl *Predict*

Description

Compute the predicted response matrix for a new data set.

Usage

```
## S3 method for class 'lsgl'
predict(object, x, sparse.data = is(x, "sparseMatrix"), ...)
```

Arguments

`object`	an object of class lsgl, produced with `lsgl`.
`x`	a data matrix of size $N_{\text{new}} \times p$.
`sparse.data`	if TRUE `x` will be treated as sparse, if `x` is a sparse matrix it will be treated as sparse by default.
`...`	ignored.

Value

Yhat — the predicted response matrix (of size $N_{\text{new}} \times K$)

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 50 #number of features
K <- 25  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)
X1<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y1 <-X1%*%B+matrix(rnorm(N*K,0,1),N,K)

X2<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y2 <-X2%*%B+matrix(rnorm(N*K,0,1),N,K)

#### Fit models using X1
lambda <- lsgl.lambda(X1, Y1, alpha = 1, d = 25L, lambda.min = 0.5, intercept = FALSE)
fit <- lsgl(X1, Y1, alpha = 1, lambda = lambda, intercept = FALSE)

# Predict Y2 using the estimated models and X2
res <- predict(fit, X2)

 # Frobenius norm \|Y2hat - Y2\|_F
sapply(res$Yhat, function(y) sqrt(sum((y - Y2)^2)))

# This is proportional to the errors compute with Err:
Err(fit, X2, Y2)*length(Y2)
```

---

print.lsgl *Print function for lsgl*

Description

This function will print some general information about the lsgl object

Usage

```
## S3 method for class 'lsgl'
print(x, ...)
```

Arguments

x	lsgl object
...	ignored

Author(s)

Martin Vincent

Examples

```
set.seed(100) # This may be removed, it ensures consistency of the daily tests

## Simulate from Y=XB+E, the dimension of Y is N x K, X is N x p, B is p x K

N <- 100 #number of samples
p <- 25 #number of features
K <- 15  #number of groups

B<-matrix(sample(c(rep(1,p*K*0.1),rep(0, p*K-as.integer(p*K*0.1)))),nrow=p,ncol=K)

X<-matrix(rnorm(N*p,1,1),nrow=N,ncol=p)
Y<-X%*%B+matrix(rnorm(N*K,0,1),N,K)

lambda<-lsgl.lambda(X,Y, alpha=1, d = 25, lambda.min= 5, intercept=FALSE)
fit <-lsgl(X,Y, alpha=1, lambda = lambda, intercept=FALSE)

# Print some information about the estimated models
fit

## Cross validation
fit.cv <- lsgl.cv(X, Y, alpha = 1, lambda = lambda, intercept = FALSE)

# Print some information
fit.cv
```

Index