

Package 'lsgl'

December 29, 2016

Type Package

Title Linear Multiple Output Sparse Group Lasso

Version 1.3.5

Date 2016-12-28

Author Martin Vincent

Maintainer Martin Vincent <martin.vincent.dk@gmail.com>

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 parallel computing for cross validation and subsampling is supported through the 'foreach' and 'doParallel' packages. Development version is on GitHub, please report package issues on GitHub.

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

BugReports <https://github.com/vincent-dk/lsgl/issues>

License GPL (>= 2)

LazyLoad yes

Imports methods, utils, stats

Depends R (>= 3.2.4), Matrix, sglOptim (== 1.3.5)

LinkingTo sglOptim, Rcpp, RcppProgress, RcppArmadillo, BH

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-12-29 01:09:02

R topics documented:

lsgl-package	2
AirlineTicketPrices	3
best_model.lsgl	3
coef.lsgl	4
cv	5
Err.lsgl	6
features.lsgl	8
features_stat.lsgl	9
fit	10
lambda	12
lsgl.algorithm.config	13
lsgl.c.config	14
lsgl.standard.config	15
models.lsgl	15
nmod.lsgl	16
parameters.lsgl	17
parameters_stat.lsgl	18
predict.lsgl	18
print.lsgl	20
subsampling	21
X	23
Y	23
Index	24

lsgl-package

Linear Multiple Output Using Sparse Group Lasso.

Description

Simultaneous feature selection and parameter estimation for linear multiple output. The algorithm finds the sparse group lasso penalized maximum likelihood estimator. This result in feature and parameter selection, and parameter estimation. Use of parallel computing for cross validation and subsampling is supported through the 'foreach' and 'doParallel' packages.

Details

The sparse gorup lasso linear mltiple output estimator is defined as

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

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 $\xi \in [0, \infty)^n$.

Author(s)

Martin Vincent <martin.vincent.dk@gmail.com>

Examples

```
# NOTE
```

```
AirlineTicketPrices  Airline Ticket Prices.
```

Description

Airline Ticket Prices.

Format

A design matrix and a response matrix

X design matrix

Y response matrix

```
best_model.lsgl      Index of best model
```

Description

Returns the index of the best model, in terms of lowest error rate

Usage

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

Arguments

object a lsgl object
... additional parameters (ignored)

Value

index of the best model.

Author(s)

Martin Vincent

coef.lsgl	<i>Nonzero Coefficients</i>
-----------	-----------------------------

Description

This function returns the nonzero coefficients (that is the nonzero entries of the *beta* matrices)

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::fit(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))
```

Description

Linear multiple output cross validation using multiple processors

Usage

```
cv(x, y, intercept = TRUE, weights = NULL, grouping = NULL,
   groupWeights = NULL, parameterWeights = NULL, alpha = 1, lambda,
   d = 100, fold = 10L, cv.indices = list(), max.threads = NULL,
   use_parallel = FALSE, algorithm.config = lsgl.standard.config)
```

Arguments

<code>x</code>	design matrix, matrix of size $N \times p$.
<code>y</code>	response matrix, matrix of size $N \times K$.
<code>intercept</code>	should the model include intercept parameters.
<code>weights</code>	sample weights, vector of size $N \times K$.
<code>grouping</code>	grouping of features, a factor or vector of length p . Each element of the factor/vector specifying the group of the feature.
<code>groupWeights</code>	the group weights, a vector of length m (the number of groups).
<code>parameterWeights</code>	a matrix of size $K \times p$.
<code>alpha</code>	the α value 0 for group lasso, 1 for lasso, between 0 and 1 gives a sparse group lasso penalty.
<code>lambda</code>	lambda.min relative to lambda.max or the lambda sequence for the regularization path.
<code>d</code>	length of lambda sequence (ignored if $\text{length}(\text{lambda}) > 1$)
<code>fold</code>	the fold of the cross validation, an integer larger than 1 and less than $N + 1$. Ignored if <code>cv.indices != NULL</code> .
<code>cv.indices</code>	a list of indices of a cross validation splitting. If <code>cv.indices = NULL</code> then a random splitting will be generated using the <code>fold</code> argument.
<code>max.threads</code>	Deprecated (will be removed in 2018), instead use <code>use_parallel = TRUE</code> and registre parallel backend (see package 'doParallel'). The maximal number of threads to be used.
<code>use_parallel</code>	If TRUE the foreach loop will use %dopar%. The user must registre the parallel backend.
<code>algorithm.config</code>	the algorithm configuration to be used.

Value

<code>Yhat</code>	the cross validation estimated response matrix
<code>Y.true</code>	the true response matrix, this is equal to the argument <code>y</code>
<code>cv.indices</code>	the cross validation splitting used
<code>features</code>	number of features used in the models
<code>parameters</code>	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 <- 50 #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)
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
fit.cv <- lsgl::cv(X1, Y1, alpha = 1, lambda = 0.1, intercept = FALSE)

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

## Do the same cross validation using 2 parallel units
cl <- makeCluster(2)
registerDoParallel(cl)

fit.cv <- lsgl::cv(X1, Y1, alpha = 1, lambda = 0.1, intercept = FALSE, use_parallel = TRUE)

stopCluster(cl)

Err(fit.cv)

```

Description

Compute and return an error for each model. The error may be spicified 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,
     loss = "RMSE", ...)
```

Arguments

object	a lsgl object.
data	a design matrix (the X matrix).
response	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::fit(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::fit(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)

```

features_stat.lsgl *Extract feature statistics*

Description

Extracts the number of nonzero features (or group) in each model.

Usage

```

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

```

Arguments

object	a lsgl object
...	ignored

Value

a vector of length `nmod(x)` or a matrix containing the number of nonzero features (or group) of the models.

Author(s)

Martin Vincent

fit

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

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

Usage

```
fit(x, y, intercept = TRUE, weights = NULL, grouping = NULL,
    groupWeights = NULL, parameterWeights = NULL, alpha = 1, lambda,
    d = 100, 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.min relative to lambda.max or the lambda sequence for the regularization path.
d	length of lambda sequence (ignored if $\text{length}(\text{lambda}) > 1$)
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 $\hat{\beta}(\lambda(1)), \dots, \hat{\beta}(\lambda(d))$ of length <code>length(return)</code> . With each entry of list holding the fitted parameters, that is matrices of size $K \times p$ (if <code>intercept = TRUE</code> 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, ensures consistency of 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)

fit <- lsgl::fit(X,Y, alpha=1, lambda = 0.1, 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(
  x = as.matrix(fit$beta[[100]]),
  main = paste("Lasso estimate (lambda =", round(fit$lambda[100], 2), ")")
)
image(solve(t(X)%*%X)%*%t(X)%*%Y, main = "Least squares estimate")

# The training error of the models
Err(fit, X, loss="OVE")
# This is simply the loss function
sqrt(N*fit$loss)

```

lambda	<i>Compute a lambda sequence for the regularization path</i>
--------	--

Description

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

Usage

```
lambda(x, y, intercept = TRUE, weights = NULL, grouping = NULL,
       groupWeights = NULL, parameterWeights = NULL, alpha = 1, d = 100L,
       lambda.min, lambda.min.rel = FALSE,
       algorithm.config = lsgl.standard.config)
```

Arguments

<code>x</code>	design matrix, matrix of size $N \times p$.
<code>y</code>	response matrix, matrix of size $N \times K$.
<code>intercept</code>	should the model include intercept parameters.
<code>weights</code>	sample weights, vector of size $N \times K$.
<code>grouping</code>	grouping of features, a factor or vector of length p . Each element of the factor/vector specifying the group of the feature.
<code>groupWeights</code>	the group weights, a vector of length m (the number of groups).
<code>parameterWeights</code>	a matrix of size $K \times p$.
<code>alpha</code>	the α value 0 for group lasso, 1 for lasso, between 0 and 1 gives a sparse group lasso penalty.
<code>d</code>	the length of lambda sequence
<code>lambda.min</code>	the smallest lambda value in the computed sequence.
<code>lambda.min.rel</code>	is lambda.min relative to lambda.max ? (i.e. actual lambda min used is <code>lambda.min*lambda.max</code> , with <code>lambda.max</code> the computed maximal lambda value)
<code>algorithm.config</code>	the algorithm configuration to be used.

Value

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

Author(s)

Martin Vincent

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_penalized_loop = TRUE,
  stepsize_opt_penalized_initial_t = 1, stepsize_opt_penalized_a = 0.1,
  stepsize_opt_penalized_b = 0.1, max_iterations_outer = 10000,
  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_penalized_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.
`max_iterations_outer`
 max iteration of outer loop
`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 <- 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)

# 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::fit(X,Y, alpha=1, lambda = lambda, intercept=FALSE, algorithm.config = config)

```

`lsgl.c.config`

Fetch information about the C side configuration of the package

Description

Fetch information about the C side configuration of the package

Usage

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

Value

list

Author(s)

Martin Vicnet

lsgl.standard.config *Standard algorithm configuration*

Description

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

Usage

```
lsgl.standard.config
```

Format

An object of class list of length 15.

Author(s)

Martin Vicnet

models.lsgl *Extract 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

Number of Models

Description

Returns the number of models used for fitting. Note that cv and subsampling objects does not containing any models even though nmod returns a positive number.

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::fit(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  $N \times K$ ,  $X$  is  $N \times p$ ,  $B$  is  $p \times K$ 

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::fit(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)
```

parameters_stat.lsgl *Extracting parameter statistics*

Description

Extracts the number of nonzero parameters in each model.

Usage

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

Arguments

object	a lsgl object
...	ignored

Value

a vector of length `nmod(x)` or a matrix containing the number of nonzero parameters of the models.

Author(s)

Martin Vincent

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::fit(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::fit(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
```

subsampling

*Subsampling***Description**

Linear multiple output subsampling using multiple processors

Usage

```
subsampling(x, y, intercept = TRUE, weights = NULL, grouping = NULL,
  groupWeights = NULL, parameterWeights = NULL, alpha = 1, lambda,
  d = 100, train, test, collapse = FALSE, max.threads = NULL,
  use_parallel = FALSE, 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.min relative to lambda.max or the lambda sequence for the regularization path (that is a vector or a list of vectors with the lambda sequence for the subsamples).
d	length of lambda sequence (ignored if $\text{length}(\text{lambda}) > 1$)
train	a list of training samples, each item of the list corresponding to a subsample. Each item in the list must be a vector with the indices of the training samples for the corresponding subsample. The length of the list must equal the length of the test list.
test	a list of test samples, each item of the list corresponding to a subsample. Each item in the list must be vector with the indices of the test samples for the corresponding subsample. The length of the list must equal the length of the training list.
collapse	if TRUE the results for each subsample will be collapse into one result (this is useful if the subsamples are not overlapping)
max.threads	Deprecated (will be removed in 2018), instead use <code>use_parallel = TRUE</code> and <code>registre parallel backend</code> (see package 'doParallel'). The maximal number of threads to be used.

`use_parallel` If TRUE the foreach loop will use `%dopar%`. The user must register the parallel backend.

`algorithm.config` the algorithm configuration to be used.

Value

`Yhat` if `collapse = FALSE` then a list of length `length(test)` containing the predicted responses for each of the test sets. If `collapse = TRUE` a list of length `length(lambda)`

`Y.true` a list of length `length(test)` containing the true responses of the test samples

`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 subsampling

train <- replicate(2, sample(1:N, 50), simplify = FALSE)
test <- lapply(train, function(idx) (1:N)[-idx])

lambda <- lapply(train, function(idx)
  lsgl::lambda(
    x = X1[idx,],
    y = Y1[idx,],
    alpha = 1,
    d = 15L,
    lambda.min = 5,
    intercept = FALSE)
)

fit.sub <- lsgl::subsampling(
  x = X1,
  y = Y1,
```

```

alpha = 1,
lambda = lambda,
train = train,
test = test,
intercept = FALSE
)

Err(fit.sub)

## Do the same cross subsampling using 2 parallel units
cl <- makeCluster(2)
registerDoParallel(cl)

# Run subsampling
# Using a lambda sequence ranging from the maximal lambda to 0.1 * maximal lambda
fit.sub <- lsgl::subsampling(
  x = X1,
  y = Y1,
  alpha = 1,
  lambda = 0.1,
  train = train,
  test = test,
  intercept = FALSE
)

stopCluster(cl)

Err(fit.sub)

```

X *Design matrix*

Description

Design matrix

Y *Response matrix*

Description

Response matrix

Index

*Topic **datasets**

lsgl.standard.config, 15

*Topic **data**

AirlineTicketPrices, 3

X, 23

Y, 23

AirlineTicketPrices, 3

best_model.lsgl, 3

coef.lsgl, 4

cv, 5

Err.lsgl, 6

features.lsgl, 8

features_stat.lsgl, 9

fit, 10

lambda, 12

lsgl-package, 2

lsgl.algorithm.config, 13

lsgl.c.config, 14

lsgl.standard.config, 15

models.lsgl, 15

nmod.lsgl, 16

parameters.lsgl, 17

parameters_stat.lsgl, 18

predict.lsgl, 18

print.lsgl, 20

subsampling, 21

X, 23

Y, 23