Package ‘sglOptim’

February 20, 2015

Type Package
Title Sparse group lasso generic optimizer
Version 1.0.122.1
Date 2014-11-04
Author Martin Vincent
Maintainer Martin Vincent <vincent@math.ku.dk>
Description Fast generic solver for sparse group lasso optimization problems. The loss (objective) function must be defined in a C++ module. This package apply template metaprogramming techniques, therefore -- when compiling the package from source -- a high level of optimization is needed to gain full speed (e.g. for the GCC compiler use -O3). Use of multiple processors for cross validation and subsampling is supported through OpenMP. The Armadillo C++ library is used as the primary linear algebra engine. (The sglOptim package version a.b.c.d is interpreted as follows: a - primary version, b - major updates and fixes, c - source revision as corresponding to R-Forge, d - minor fixes made only to the CRAN branch of the source)

URL http://dx.doi.org/10.1016/j.csda.2013.06.004
License GPL (>= 2)
LazyLoad yes
Depends R (>= 3.0.0), Matrix,
LinkingTo Rcpp, RcppProgress, RcppArmadillo, BH
Collate 'lambda_sequence.R' 'prepare_args.R' 'sgl_fit.R'
   'sgl_config.R' 'sgl_predict.R' 'sgl_cv.R' 'sgl_subsampling.R'
   'sgl_navigate.R'
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-11-04 20:11:16
R topics documented:

- coef.sgl
- compute_error
- create.sgldata
- Err
- features
- features.sgl
- models
- models.sgl
- nmod
- nmod.sgl
- parameters
- parameters.sgl
- prepare.args
- prepare.args.sgldata
- rearrange
- rearrange.sgldata
- sgl.algorithm.config
- sgl.standard.config
- sgl_cv
- sgl_fit
- sgl_lambda_sequence
- sgl_predict
- sgl_print
- sgl_subsampling
- test.data

Index

| coef.sgl | Extracting the nonzero coefficients |

Description

Extracting the nonzero coefficients

Usage

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

Arguments

- object: a sgl object
- index: indices of the models
- ...: not used
compute_error

Value

a list of with nonzero coefficients of the models

Author(s)

Martin Vincent

compute_error      Helper function for computing error rates

Description

Helper function for computing error rates

Usage

compute_error(object, data = NULL, response.name,
               response, loss)

Arguments

object      a object containing responses
data       a data object
response.name the name of the response, if response.name = NULL then x will be treated as the response.
response     the response
loss         the loss function

Details

This function can be used to compute error rates. It is consist with the use cases of the Err genetic function. (see msgl package for an example of how to use this function)

Value

a vector with the computed error rates

Author(s)

Martin Vincent
create.sgldata  

Create a sgldata object

Description

Creates a sgldata object from a design matrix and an optional response vector or matrix.

Usage

\[
\text{create.sgldata}(x, y, \text{weights} = \text{rep}(1/nrow(x), nrow(x)), \\
\text{sampleGrouping} = \text{NULL}, \text{group.names} = \text{NULL}, \\
\text{sparseX} = \text{is}(x, \text{"sparseMatrix"}))
\]

Arguments

- \text{x}: the design matrix, a matrix of size \( N \times p \) (will be parsed to the loss module as \( X \)).
- \text{y}: the responses, \text{NULL}, a vector or a matrix (will be parsed to the loss module as \( Y \)).
- \text{weights}: sample weights, a vector of length \( N \) (will be parsed to the loss module as \( w \)).
- \text{sampleGrouping}: grouping of samples, a factor of length \( N \) (will be parsed to the loss module as \( G \)). Default is no grouping (NULL), that is all samples is the same group.
- \text{group.names}: a vector with the names of the parameter groups (the length must equal the number of rows in the \( \beta \) matrix).
- \text{sparseX}: if \text{TRUE} \text{x} will be treated as sparse, if \text{FALSE} \text{x} will be treated as dens.

Author(s)

Martin Vincent

See Also

Other sgldata: prepare.args, prepare.args.sgldata, rearrange.sgldata

Err  

Generic function for computing error rates

Description

Compute and returns an error rate for each model contained in \text{x}. See details for generic use cases.

Usage

\[
\text{Err}(\text{object, data, response, ...})
\]
Arguments

- **object**: an object
- **data**: a data object
- **response**: a response object
- ... additional parameters (optional)

Details

The following generic use case should be supported (see for example `msgl` package for an implementation):

1. With `fit` a sgl fit object with models estimated using `x` data, the code
   \[ \text{Err}(\text{fit}, \text{x}) \]
   should return a vector with the *training errors* of the models.

2. With `x.new` a new data set with known responses `response.new`, the code
   \[ \text{Err}(\text{fit}, \text{x.new}, \text{response.new}) \]
   should return a vector with the errors of the models when applied to the new data set.

3. With `fit.cv` a sgl cross validation object, the code
   \[ \text{Err}(\text{fit.cv}) \]
   should return a vector with estimates of the *expected generalization errors* of the models (i.e. the cross validation errors).

4. If subsampling is supported then, with `fit.sub` a sgl subsampling object, the code
   \[ \text{Err}(\text{fit.sub}) \]
   should return a matrix with the test errors (each column corresponding to a model, i.e. rows corresponds to tests).

Value

A vector of length `nmod(object)` or a matrix with `nmod(object)` columns containing error rates for the models.

Author(s)

Martin Vincent

See Also

`compute_error`
features

Generic function for extracting nonzero features (or groups)

Description

Extracts nonzero features for each model.

Usage

features(object, ...)

Arguments

object an object
... additional parameters (optional)

Value

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

Author(s)

Martin Vincent

features.sgl

Extracting nonzero features

Description

Extracting nonzero features

Usage

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

Arguments

object a sgl object
... not used

Value

a list of vectors containing the nonzero features (that is nonzero columns of the \emph{beta} matrices)

Author(s)

Martin Vincent
models

Generic function for extracting the fitted models

Description

Returns the fitted models

Usage

models(object, index, ...)

Arguments

object an object
index a vector of indices of the models to be returned
... additional parameters (optional)

Value

a list of length length(index) containing the models

Author(s)

Martin Vincent

models.sgl

Returns the estimated models (that is the beta matrices)

Description

Returns the estimated models (that is the beta matrices)

Usage

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

Arguments

object a sgl object
index indices of the models to be returned
... not used
Value

a list of sparse matrices

Author(s)

Martin Vincent

nmod

Generic function for counting the number of models

Description

Returns the number of models

Usage

nmod(object, ...)

Arguments

object an object
... additional parameters (optional)

Value

the number of models contained in the object x.

Author(s)

Martin Vincent

nmod.sgl

Returns the number of models in a sgl object

Description

Returns the number of models in a sgl object

Usage

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

parameters

Arguments

object       a sgl object
...          not used

Value

the number of models in object

Author(s)

Martin Vincent

parameters  Generic function for extracting nonzero parameters

Description

Extracts nonzero parameters for each model.

Usage

parameters(object, ...)

Arguments

object       an object
...          additional parameters (optional)

Value

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

Author(s)

Martin Vincent
parameters.sgl  Extracting nonzero parameters

Description
Extracting nonzero parameters

Usage
```r
## S3 method for class 'sgl'
parameters(object, ...)
```

Arguments
- **object**: a sgl object
- **...**: not used

Value
A list of vectors containing the nonzero parameters (that is nonzero entries of the beta matrices)

Author(s)
Martin Vincent

prepare.args  Generic function for preparing the sgl call arguments

Description
Compute and prepare the sgl call arguments for the objective function

\[
\text{loss}(\text{data})(\beta) + \lambda \left( (1 - \alpha) \sum_{J=1}^{m} \gamma_J \| \beta^{(J)} \|_2 + \alpha \sum_{i=1}^{n} \xi_i |\beta_i| \right)
\]

where loss is a loss/objective function. The \( n \) parameters are organized in the parameter matrix \( \beta \) with dimension \( q \times p \). The vector \( \beta^{(J)} \) denotes the \( J \) parameter group, the dimension of \( \beta^{(J)} \) is denote by \( d_J \). The dimensions \( d_J \) must be multiple of \( q \), and \( \beta = (\beta^{(1)} \ldots \beta^{(m)}) \). The group weights \( \gamma \in [0, \infty)^m \) and the parameter weights \( \xi \in [0, \infty)^{qp} \).

Usage
```r
prepare.args(data, ...)
```
Arguments

data a data object
... additional parameters

Value

block.dim a vector of length \( m \), containing the dimensions \( d_j \) of the groups (i.e. the number of parameters in the groups)
groupWeights a vector of length \( m \), containing the group weights
parameterWeights a matrix of dimension \( q \times p \), containing the parameter weights
alpha the \( \alpha \) value
data the data parsed to the loss module
group.order original order of the columns of \( \beta \). Before sgl routines return \( \beta \) will be reorganized according to this order.

Author(s)

Martin Vincent

See Also

prepare.args.sgldata

Other sgldata: create.sgldata, prepare.args.sgldata, rearrange.sgldata

Description

Prepare sgl function arguments using sgldata.

Usage

```r
## S3 method for class 'sgldata'
prepare.args(data, parameterGrouping,
             groupWeights, parameterWeights, alpha, ...)
```
Arguments

- **data**: a sgldata object
- **parameterGrouping**: grouping of parameters, a vector of length \( p \). Each element of the vector specifying the group of the parameters in the corresponding column of \( \beta \).
- **groupWeights**: the group weights, a vector of length \( \text{length}(\text{unique}(\text{parameterGrouping})) \) (the number of groups).
- **parameterWeights**: a matrix of size \( q \times p \), that is the same dimension as \( \beta \).
- **alpha**: the \( \alpha \) value 0 for group lasso, 1 for lasso, between 0 and 1 gives a sparse group lasso penalty.
- **...**: not used

Author(s)

Martin Vincent

See Also

Other sgldata: `create.sgldata`, `prepare.args`, `rearrange.sgldata`

---

**rearrange**

*Generic rearrange function*

### Description

Rearrange the order of the covariates in the data object.

### Usage

```r
rearrange(data, covariate.order, ...)
```

### Arguments

- **data**: a data object
- **covariate.order**: the new order of the covariates
- **...**: additional parameters

### Value

- a rearranged data object of same class as data

### Author(s)

Martin Vincent
Description

Rearrange the order of the covariates in a sgldata object.

Usage

```r
## S3 method for class 'sgldata'
rearrange(data, covariate.order, ...)
```

Arguments

- `data`: a sgldata object
- `covariate.order`: the new order of the covariates
- `...`: not used

Value

A sgldata object with the covariates reordered

Author(s)

Martin Vincent

See Also

Other sgldata: `create.sgldata`, `prepare.args`, `prepare.args.sgldata`
Create a new algorithm configuration

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

Usage
```
sgl.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 = TRUE,
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, 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.
**sgl.standard.config**

- **stepsize_opt_penalized_b**
  - step-size optimization parameter.
- **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**

```r
config.no_progressbar <- sgl.algorithm.config(verbise = FALSE)
```

---

**Description**

```r
sgl.standard.config <- sgl.algorithm.config()
```

**Usage**

```r
sgl.standard.config
```

**Format**

List of 13
- `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_middle_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 TRUE
- `use_stepsize_optimization_in_penalized_loop`: logi TRUE
- `stepsize_opt_penalized_initial_t`: num 1
- `stepsize_opt_penalized_a`: num 0.1
- `stepsize_opt_penalized_b`: num 0.1
- `verbose`: logi TRUE

**Author(s)**

Martin Vincent
Generic sparse group lasso cross validation using multiple possessors

Usage

\[
\text{sgl_cv}(\text{module name}, \text{PACKAGE}, \text{data}, \text{parameterGrouping}, \text{groupWeights, parameterWeights, alpha, lambda, fold = 2, cv.indices = list(), max.threads = 2, algorithm.config = sgl.standard.config})
\]

Arguments

- `module_name`: reference to objective specific C++ routines.
- `PACKAGE`: name of the calling package.
- `data`: a list of data objects – will be parsed to the specified module.
- `parameterGrouping`: grouping of parameters, a vector of length \( p \). Each element of the vector specifying the group of the parameters in the corresponding column of \( \beta \).
- `groupWeights`: the group weights, a vector of length \( \text{length(unique(parameterGrouping))} \) (the number of groups).
- `parameterWeights`: a matrix of size \( q \times p \).
- `alpha`: the \( \alpha \) 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. If `fold` ≤ `max(table(classes))` then the data will be split into `fold` disjoint subsets keeping the ration of classes approximately equal. Otherwise the data will be split into `fold` disjoint subsets without keeping the ration fixed.
- `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

- **responses**: content will depend on the C++ response class
- **cv.indices**: the cross validation splitting used
- **features**: number of features used in the models
- **parameters**: number of parameters used in the models
- **lambda**: the lambda sequence used.

**Author(s)**

Martin Vincent

---

**sgl_fit**

*Fit a sparse group lasso regularization path.*

**Description**

A sequence of minimizers (one for each lambda given in the lambda argument) of

\[ \text{loss}(\beta) + \lambda \left( (1 - \alpha) \sum_{J=1}^{m} \gamma_J \|\beta^{(J)}\|_2 + \alpha \sum_{i=1}^{n} \xi_i |\beta_i| \right) \]

where \( \text{loss} \) is the loss/objective function specified by `module_name`. The parameters are organized in the parameter matrix \( \beta \) with dimension \( q \times p \). The vector \( \beta^{(J)} \) denotes the \( J \) parameter group. The group weights \( \gamma \in (0, \infty)^m \) and the parameter weights \( \xi = (\xi^{(1)}, \ldots, \xi^{(m)}) \in [0, \infty)^n \) with \( \xi^{(1)} \in [0, \infty)^{n_1}, \ldots, \xi^{(m)} \in [0, \infty)^{n_m} \).

**Usage**

```r
sgl_fit(module_name, PACKAGE, data, parameterGrouping, groupWeights, parameterWeights, alpha, lambda, return = 1:length(lambda), algorithm.config = sgl.standard.config)
```

**Arguments**

- **module_name**: reference to objective specific C++ routines.
- **PACKAGE**: name of the calling package.
- **data**: a list of data objects – will be parsed to the specified module.
- **parameterGrouping**: grouping of parameters, a vector of length \( p \). Each element of the vector specifying the group of the parameters in the corresponding column of \( \beta \).
- **groupWeights**: the group weights, a vector of length \( \text{length(unique(parameterGrouping))} \) (the number of groups).
- **parameterWeights**: a matrix of size \( q \times p \).
alpha the $\alpha$ 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.

return the lambda sequence for the regularization path.

algorithm.config the algorithm configuration to be used.

Value

beta the fitted parameters – a list of length length(return) with each entry a matrix of size $q \times (p + 1)$ holding the fitted parameters.

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

Description

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

Usage

sgl_lambda_sequence(module_name, PACKAGE, data, parameterGrouping, groupWeights, parameterWeights,
alpha = 0.5, d = 100, lambda.min,
algorithm.config = sgl.standard.config)

Arguments

module_name reference to objective specific C++ routines.

PACKAGE name of the calling package.

data a list of data objects – will be parsed to the specified module.

parameterGrouping grouping of parameters, a vector of length p. Each element of the vector specifying the group of the parameters in the corresponding column of $\beta$.

groupWeights the group weights, a vector of length length(unique(parameterGrouping)) (the number of groups).
parameterWeights
  a matrix of size $q \times p$.
alpha
  the $\alpha$ 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

---

**Description**

Sgl predict

**Usage**

```r
sgl_predict(module_name, PACKAGE, object, data, ...)
```

**Arguments**

- `module_name`: reference to objective specific C++ routines.
- `PACKAGE`: name of the calling package.
- `object`: a sgl object containing a list of estimated models.
- `data`: a list of data objects – will be parsed to the specified module.
- `...`: not used.

**Value**

- `responses`: content will depend on the C++ response class
- `lambda`: the lambda sequence used.

**Author(s)**

  Martin Vincent
sgl_print  
*Print information about sgl object*

**Description**

Print information about sgl object

**Usage**

```
sgl_print(x)
```

**Arguments**

- **x**: a object of sgl family class

**Author(s)**

Martin Vincent

---

sgl_subsampling  
*Generic sparse group lasso subsampling procedure*

**Description**

Support the use of multiple processors.

**Usage**

```
sgl_subsampling(module_name, PACKAGE, data,
    parameterGrouping, groupWeights, parameterWeights,
    alpha, lambda, training, test, collapse = FALSE,
    max.threads = 2,
    algorithm.config = sgl.standard.config)
```

**Arguments**

- **module_name**: reference to objective specific C++ routines.
- **PACKAGE**: name of the calling package.
- **data**: a list of data objects – will be parsed to the specified module.
- **parameterGrouping**: grouping of parameters, a vector of length \( p \). Each element of the vector specifying the group of the parameters in the corresponding column of \( \beta \).
- **groupWeights**: the group weights, a vector of length \( \text{length(unique(parameterGrouping))} \) (the number of groups).
parameterWeights

- a matrix of size \( q \times p \).

alpha

- the \( \alpha \) 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.

training

- 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

- the maximal number of threads to be used.

algorithm.config

- the algorithm configuration to be used.

Value

- responses: content will depend on the C++ response class
- features: number of features used in the models
- parameters: number of parameters used in the models
- lambda: the lambda sequence used.

Author(s)

- Martin Vincent

---

test.data

*Simulated data set*

Description

This data set is for testing only.
Index

*Topic **datasets**
  - sgl.standard.config, 15
  - test.data, 21

*Topic **data**
  - test.data, 21

  - coef.sgl, 2
  - compute_error, 3
  - create.sgldata, 4, 11–13

Err, 4

- features, 6
- features.sgl, 6

- models, 7
- models.sgl, 7

- nmod, 8
- nmod.sgl, 8

- parameters, 9
- parameters.sgl, 10
- prepare.args, 4, 10, 12, 13
- prepare.args.sgldata, 4, 11, 11, 13

rearrange, 12
rearrange.sgldata, 4, 11, 12, 13

- sgl.algorithm.config, 14
- sgl.standard.config, 15
- sgl_cv, 16
- sgl_fit, 17
- sgl_lambda_sequence, 18
- sgl_predict, 19
- sgl_print, 20
- sgl_subsampling, 20

- test.data, 21