Package ‘grpSLOPE’

April 7, 2020

Type Package
Title Group Sorted L1 Penalized Estimation
Version 0.3.0
Description Group SLOPE is a penalized linear regression method that is used for adaptive selection of groups of significant predictors in a high-dimensional linear model. The Group SLOPE method can control the (group) false discovery rate at a user-specified level (i.e., control the expected proportion of irrelevant among all selected groups of predictors).
License GPL-3
URL https://github.com/agisga/grpSLOPE.git
BugReports https://github.com/agisga/grpSLOPE/issues
LazyData TRUE
LinkingTo Rcpp
Imports Rcpp
RoxygenNote 6.1.1
Suggests testthat, knitr, rmarkdown, pander, isotone
VignetteBuilder knitr
NeedsCompilation yes
Encoding UTF-8
Author Alexej Gossmann [aut, cre], Damian Brzyski [aut], Weijie Su [aut], Małgorzata Bogdan [aut], Ewout van den Berg [ctb] (Code adapted from 'SLOPE' version 0.1.3, as well as from http://statweb.stanford.edu/~candes/SortedL1/software.html under GNU GPL-3), Emmanuel Candes [ctb] (Code adapted from 'SLOPE' version 0.1.3, as well as from http://statweb.stanford.edu/~candes/SortedL1/software.html under GNU GPL-3),
admmSolverGroupSLOPE

Alternating direction method of multipliers

Description

Compute the coefficient estimates for the Group SLOPE problem.

Usage

admmSolverGroupSLOPE(y, A, group, wt, lambda, rho = NULL,
max.iter = 10000, verbose = FALSE, absolute.tol = 1e-04,
relative.tol = 1e-04, z.init = NULL, u.init = NULL, ...)

Arguments

y the response vector
A the model matrix
group A vector describing the grouping structure. It should contain a group id for each predictor variable.
wt A vector of weights (per coefficient)
lambda A decreasing sequence of regularization parameters \( \lambda \)
rho Penalty parameter in the augmented Lagrangian (see Boyd et al., 2011)
max.iter Maximal number of iterations to carry out
verbose A logical specifying whether to print output or not
absolute.tol The absolute tolerance used in the stopping criteria for the primal and dual feasibility conditions (see Boyd et al., 2011, Sec. 3.3.1)
relative.tol The relative tolerance used in the stopping criteria for the primal and dual feasibility conditions (see Boyd et al., 2011, Sec. 3.3.1)
z.init An optional initial value for the iterative algorithm
u.init An optional initial value for the iterative algorithm
...
Options passed to \texttt{prox\_sorted\_L1}

Details

\texttt{admmSolverGroupSLOPE} computes the coefficient estimates for the Group SLOPE model. The employed optimization algorithm is the alternating direction method of multipliers (ADMM).

Value

A list with the entries:

\begin{itemize}
  \item \texttt{x} Solution (n-by-1 matrix)
  \item \texttt{status} Convergence status: 1 if optimal, 2 if iteration limit reached
  \item \texttt{iter} Number of iterations of the ADMM method
\end{itemize}

References


Examples

```r
set.seed(1)
A <- matrix(runif(100, 0, 1), 10, 10)
grp <- c(0, 0, 1, 1, 2, 2, 2, 2, 2, 3)
wt <- c(2, 2, 2, 2, 5, 5, 5, 5, 5, 1)
x <- c(0, 0, 5, 1, 0, 0, 0, 1, 0, 3)
y <- A %*% x
lam <- 0.1 * (10:7)
result <- admmSolverGroupSLOPE(y = y, A = A, group = grp, wt = wt, 
lambda=lam, rho = 1, verbose = FALSE)
result$x
# [1,] 0.000000
# [2,] 0.000000
```
# Extract model coefficients

## S3 method for class `grpSLOPE`

```r
coef(object, scaled = TRUE, ...)
```

### Arguments

- **object**: A `grpSLOPE` object
- **scaled**: Should the coefficients be returned for the normalized version of the design matrix?
- **...**: Potentially further arguments passed to and from methods

### Details

If `scaled` is set to `TRUE`, then the coefficients are returned for the normalized version of the design matrix, which is the scale on which they were computed. If `scaled` is set to `FALSE`, then the coefficients are transformed to correspond to the original (unaltered) design matrix. In case that `scaled = FALSE`, an estimate for the intercept term is returned with the other coefficients. In case that `scaled = TRUE`, the estimate of the intercept is always equal to zero, and is not explicitly provided.

### Value

A named vector of regression coefficients where the names signify the group that each entry belongs to.
getGroupID

Examples

```r
set.seed(1)
A <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(letters[1:20]), each = 5)
b <- c(rep(1, 20), rep(0, 80))
y <- A %*% b + rnorm(10)
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1)
head(coef(result), 8)
#  a_1  a_2  a_3  a_4  a_5  b_1  b_2  b_3
#  7.942177 7.979269 8.667013 8.514861 10.026664 8.963364 10.037355 10.448692
head(coef(result, scaled = FALSE), 8)
# (Intercept)  a_1  a_2  a_3  a_4  a_5  b_1  b_2
# -0.4418113 0.8886878 0.8372108 0.8422089 0.8629597 0.8615827 0.9323849 0.9333445
```

getGroupID

Get a groupID object

Description

Mostly intended for internal use.

Usage

getGroupID(group)

Arguments

- **group**: A vector describing the grouping structure. It should contain a group id for each predictor variable.

Value

An object of class groupID, which is a list, whose members are vectors of indices corresponding to each group. The names of the list members are the corresponding group names.

Examples

```r
group <- c("A", "A", 2, 9, "A", 9, 2, "A")
group.id <- getGroupID(group)
group.id
# $A
# [1] 1 2 5 9
# $'2'
# [1] 3 8
# $'9'
# [1] 4 6 7
```
# attr("class")
# [1] "groupID"

grpSLOPE

**Group SLOPE (Group Sorted L-One Penalized Estimation)**

---

**Description**

Performs selection of significant groups of predictors and estimation of the corresponding coefficients using the Group SLOPE method (see Brzyski et. al., 2016).

**Usage**

```r
grpSLOPE(X, y, group, fdr, lambda = "corrected", sigma = NULL,
          verbose = FALSE, orthogonalize = NULL, normalize = TRUE,
          max.iter = 10000, dual.gap.tol = 1e-06, infeas.tol = 1e-06,
          x.init = NULL, ...)
```

**Arguments**

- **X**  
The model matrix
- **y**  
The response variable
- **group**  
A vector describing the grouping structure. It should contain a group id for each predictor variable.
- **fdr**  
Target group false discovery rate (gFDR)
- **lambda**  
Method used to obtain the regularizing sequence lambda. Possible values are "max", "mean", and "corrected" (default). See `lambdaGroupSLOPE` for detail. Alternatively, any non-increasing sequence of the correct length can be passed.
- **sigma**  
Noise level. If omitted, estimated from the data, using Procedure 2 in Brzyski et. al. (2016).
- **verbose**  
A logical specifying whether to print output or not
- **orthogonalize**  
Whether to orthogonalize the model matrix within each group. Do not set manually unless you are certain that your data is appropriately pre-processed.
- **normalize**  
Whether to center the input data and re-scale the columns of the design matrix to have unit norms. Do not disable this unless you are certain that your data are appropriately pre-processed.
- **max.iter**  
See `proximalGradientSolverGroupSLOPE`.
- **dual.gap.tol**  
See `proximalGradientSolverGroupSLOPE`.
- **infeas.tol**  
See `proximalGradientSolverGroupSLOPE`.
- **x.init**  
See `proximalGradientSolverGroupSLOPE`.
- **...**  
Options passed to `prox_sorted_L1`
Details

Multiple methods are available to generate the regularizing sequence \( \lambda \), see \( \text{lambdaGroupSLOPE} \) for detail. The model matrix is transformed by orthogonalization within each group (see Section 2.1 in Brzyski et al., 2016), and penalization is imposed on \( \| X_I \beta_I \| \). When \( \text{orthogonalize} = \text{TRUE} \), due to within group orthogonalization, the solution vector \( \beta \) cannot be computed, if a group submatrix does not have full column rank (e.g., if there are more predictors in a selected group than there are observations). In that case only the solution vector \( c \) of the transformed (orthogonalized) model is returned. Additionally, in any case the vector \( \text{group.norms} \) is returned with its \( i \)th entry being \( \| X_I \beta_I \| \), i.e., the overall effect of each group. Note that all of these results are returned on the scale of the normalized versions of \( X \) and \( y \). However, \( \text{original.scale} \) contains the regression coefficients transformed to correspond to the original (unaltered) \( X \) and \( y \). In that case, an estimate for the intercept term is also returned with the other coefficients in \( \text{original.scale} \) (while on the normalized scale the estimate of the intercept is always equal to zero, and is not explicitly provided in the \( \text{grpSLOPE} \) output).

Value

A list with members:

- \( \text{beta} \) Solution vector. See Details.
- \( c \) Solution vector of the transformed model. See Details.
- \( \text{group.norms} \) Overall effect of each group. See Details.
- \( \text{selected} \) Names of selected groups (i.e., groups of predictors with at least one non-zero coefficient estimate)
- \( \text{optimal} \) Convergence status
- \( \text{iter} \) Iterations of the proximal gradient method
- \( \lambda \) Regularizing sequence
- \( \text{lambda.method} \) Method used to construct the regularizing sequence
- \( \sigma \) (Estimated) noise level
- \( \text{group} \) The provided grouping structure (corresponding to \( \beta \))
- \( \text{group.c} \) Grouping structure of the transformed model (corresponding to \( c \))
- \( \text{original.scale} \) A list containing the estimated intercept and regression coefficients on the original scale. See Details.

References


Examples

```r
# generate some data
set.seed(1)
A <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(1:20), each=5)
b <- c(runif(20), rep(0, 80))
# (i.e., groups 1, 2, 3, 4, are truly significant)
y <- A %*% b + rnorm(10)
fdr <- 0.1 # target false discovery rate
# fit a Group SLOPE model
result <- grpSLOPE(X=A, y=y, group=grp, fdr=fdr)
result$selected
# [1] "1" "2" "3" "4" "14"
result$sigma
# [1] 0.7968632
head(result$group.norms)
#         1         2         3         4         5         6
# 2.905449 5.516103 8.964201 10.253792 0.000000 0.000000
```

---

**lambdaGroupSLOPE**

Regularizing sequence for Group SLOPE

Description

Generate the regularizing sequence lambda for the Group SLOPE problem according to one of multiple methods (see Details).

Usage

`lambdaGroupSLOPE(method, fdr, group, wt, n.obs = NULL)`

Arguments

- `method` Possible values are "max", "mean", and "corrected". See under Details.
- `fdr` Target group false discovery rate (gFDR)
- `group` A vector describing the grouping structure. It should contain a group id for each predictor variable.
- `wt` A named vector of weights, one weight per group of predictors (named according to names as in vector `group`)
- `n.obs` Number of observations (i.e., number of rows in `A`); required only if method is "corrected"
Multiple methods are available to generate the regularizing sequence \(\lambda\):

- "max" – lambdas as in Theorem 2.5 in Brzyski et al. (2016). Provalby controls gFDR in orthogonal designs.
- "mean" – lambdas of equation (2.16) in Brzyski et al. (2016). Applicable for gFDR control in orthogonal designs. Less conservative than "max".
- "corrected" – lambdas of Procedure 1 in Brzyski et al. (2016); in the special case that all group sizes are equal and \(wt\) is a constant vector, Procedure 6 of Brzyski et al. (2016) is applied. Applicable for gFDR control when predictors from different groups are stochastically independent.

References


Examples

```r
# specify 6 groups of sizes 2, 3, and 4
group <- c(1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 5, 5, 5, 6, 6, 6, 6)

# set the weight for each group to the square root of the group's size
wt <- rep(c(sqrt(2), sqrt(3), sqrt(4)), 2)
names(wt) <- 1:6

# compute different lambda sequences
lambda.max <- lambdaGroupSLOPE(method="max", fdr=0.1, group=group, wt=wt)
lambda.mean <- lambdaGroupSLOPE(method="mean", fdr=0.1, group=group, wt=wt)
lambda.corrected <- lambdaGroupSLOPE(method="corrected", fdr=0.1, 
                               group=group, wt=wt, n.obs=1000)

rbind(lambda.max, lambda.mean, lambda.corrected)
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.023449</td>
<td>1.844234</td>
<td>1.730818</td>
<td>1.645615</td>
<td>1.576359</td>
<td>1.517427</td>
</tr>
<tr>
<td>1.880540</td>
<td>1.723559</td>
<td>1.626517</td>
<td>1.554561</td>
<td>1.496603</td>
<td>1.447609</td>
</tr>
<tr>
<td>1.880540</td>
<td>1.729811</td>
<td>1.637290</td>
<td>1.568971</td>
<td>1.514028</td>
<td>1.467551</td>
</tr>
</tbody>
</table>

**predict.grpSLOPE**

Obtain predictions

Obtain predictions from a grpSLOPE model on new data
Usage

## S3 method for class 'grpSLOPE'
predict(object, newdata, ...)

Arguments

object A grpSLOPE object
newdata Predictor variables arranged in a matrix
... Potentially further arguments passed to and from methods

Details

Note that newdata must have the same shape, and must contain the same predictor variables as columns in the same order as the design matrix $X$ that was used for the grpSLOPE model fit.

Examples

```r
set.seed(1)
A <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(1:20), each = 5)
b <- c(rep(1, 20), rep(0, 80))
y <- A %*% b + rnorm(10)
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1)
newdata <- matrix(rnorm(800), 8, 100)
# group SLOPE predictions:
predict(result, newdata)
# [1] -5.283385 -6.313938 -3.173068 1.901488 9.796677 -0.144516 -0.611164 -5.167620
# true mean values:
as.vector(newdata %*% b)
# [1] -5.0937160 -6.5814111 -3.5776124 2.7877449 11.0668777 1.0253236 -0.4261076 -4.8622940
```

proxGroupSortedL1

Prox for group SLOPE

Description

Evaluate the proximal mapping for the group SLOPE problem.

Usage

proxGroupSortedL1(y, group, lambda, ...)
Arguments

\( y \) The response vector

\( \text{group} \) Either a vector or an object of class groupID (e.g., as produced by \text{getGroupID}), which is describing the grouping structure. If it is a vector, then it should contain a group id for each predictor variable.

\( \lambda \) A decreasing sequence of regularization parameters \( \lambda \)

\( \ldots \) Options passed to \text{prox\_sorted\_L1}

Details

\text{proxGroupSortedL1} evaluates the proximal mapping of the group SLOPE problem by reducing it to the prox for the (regular) SLOPE and then applying the fast prox algorithm for the Sorted L1 norm.

References


Examples

```r
gp <- c(0,0,0,1,1,0,2,1,0,2)
proxGroupSortedL1(y = 1:10, group = gp, lambda = c(10, 9, 8))
# [1] 0.2032270 0.4064540 0.6096810 0.8771198 1.0963997 1.2193620 1.3338960
# [8] 1.7542395 1.8290430 1.9055657
```

---

\textit{proximalGradientSolverGroupSLOPE}

\textit{Proximal gradient method for Group SLOPE}

Description

Compute the coefficient estimates for the Group SLOPE problem.

Usage

```r
proximalGradientSolverGroupSLOPE(y, A, group, wt, lambda, max.iter = 10000, verbose = FALSE, dual.gap.tol = 1e-06, infeas.tol = 1e-06, x.init = NULL, \ldots)
```
Arguments

- **y**  
  the response vector

- **A**  
  the model matrix

- **group**  
  A vector describing the grouping structure. It should contain a group id for each predictor variable.

- **wt**  
  A vector of weights (per coefficient)

- **lambda**  
  A decreasing sequence of regularization parameters \( \lambda \)

- **max.iter**  
  Maximal number of iterations to carry out

- **verbose**  
  A logical specifying whether to print output or not

- **dual.gap.tol**  
  The tolerance used in the stopping criteria for the duality gap

- **infeas.tol**  
  The tolerance used in the stopping criteria for the infeasibility

- **x.init**  
  An optional initial value for the iterative algorithm

- **...**  
  Options passed to `prox_sorted_L1`

Details

`proximalGradientSolverGroupSLOPE` computes the coefficient estimates for the Group SLOPE model. The employed optimization algorithm is FISTA with backtracking Lipschitz search.

Value

A list with the entries:

- **x**  
  Solution (n-by-1 matrix)

- **status**  
  Convergence status: 1 if optimal, 2 if iteration limit reached

- **L**  
  Approximation of the Lipschitz constant (step size)

- **iter**  
  Iterations of the proximal gradient method

- **L.iter**  
  Total number of iterations spent in Lipschitz search

References


prox_sorted_L1

Examples

```r
set.seed(1)
A <- matrix(runif(100, 0, 1), 10, 10)
grp <- c(0, 0, 1, 1, 2, 2, 2, 2, 2, 3)
wt <- c(2, 2, 2, 2, 5, 5, 5, 5, 5, 1)
x <- c(0, 0, 5, 1, 0, 0, 0, 1, 0, 3)
y <- A %*% x
lam <- 0.1 * (10:7)
result <- proximalGradientSolverGroupSLOPE(y=y, A=A, group=grp, wt=wt, lambda=lam, verbose=FALSE)
result$x
# [,1]
# [1,] 0.000000
# [2,] 0.000000
# [3,] 3.856005
# [4,] 2.080736
# [5,] 0.000000
# [6,] 0.000000
# [7,] 0.000000
# [8,] 0.000000
# [9,] 0.000000
# [10,] 3.512833
```

prox_sorted_L1  Prox for sorted L1 norm

Description

Compute the prox for the sorted L1 norm. That is, given a vector \(x\) and a decreasing vector \(\lambda\), compute the unique value of \(y\) minimizing

\[
\frac{1}{2} \|x - y\|_2^2 + \sum_{i=1}^{n} \lambda_i |x_{(i)}|.
\]

At present, two methods for computing the sorted L1 prox are supported. By default, we use a fast custom C implementation. Since SLOPE can be viewed as an isotonic regression problem, the prox can also be computed using the isotone package. This option is provided primarily for testing.

Usage

```r
prox_sorted_L1(x, lambda, method = c("c", "isotone"))
```

Arguments

- `x`  input vector
- `lambda`  vector of \(\lambda\)'s, sorted in decreasing order
- `method`  underlying prox implementation, either 'c' or 'isotone' (see Details)
Details

This function has been adapted (with only cosmetic changes) from the R package SLOPE version 0.1.3, due to this function being deprecated and defunct in SLOPE versions which are newer than 0.1.3.

sigma

Extract (estimated) noise level

Description

Extract the noise level of the grpSLOPE model.

Usage

## S3 method for class 'grpSLOPE'
sigma(object, ...)

Arguments

object
A grpSLOPE object

... Potentially further arguments passed to and from methods

Details

This basically obtains object$sigma. For R (>= 3.3.0) sigma is an S3 method with the default method coming from the stats package.

Examples

set.seed(1)
A <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(1:20), each = 5)
b <- c(rep(1, 20), rep(0, 80))
y <- A %*% b + rnorm(10)
# model with unknown noise level
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1)
sigma(result)
# [1] 0.6505558
# model with known noise level
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1, sigma = 1)
sigma(result)
# [1] 1
SLOPE_solver

Sorted L1 solver

Description

Solves the sorted L1 penalized regression problem: given a matrix \( A \), a vector \( b \), and a decreasing vector \( \lambda \), find the vector \( x \) minimizing

\[
\frac{1}{2} \|Ax - b\|_2^2 + \sum_{i=1}^{p} \lambda_i |x|_{(i)}.
\]

This optimization problem is convex and is solved using an accelerated proximal gradient descent method.

Usage

SLOPE_solver(A, b, lambda, initial = NULL, prox = prox_sorted_L1, max_iter = 10000, grad_iter = 20, opt_iter = 1, tol_infeas = 1e-06, tol_rel_gap = 1e-06)

Arguments

- \( A \) a \( n \)-by-\( p \) matrix
- \( b \) vector of length \( n \)
- \( \lambda \) vector of length \( p \), sorted in decreasing order
- \( \text{initial} \) initial guess for \( x \)
- \( \text{prox} \) function that computes the sorted L1 prox
- \( \text{max_iter} \) maximum number of iterations in the gradient descent
- \( \text{grad_iter} \) number of iterations between gradient updates
- \( \text{opt_iter} \) number of iterations between checks for optimality
- \( \text{tol_infeas} \) tolerance for infeasibility
- \( \text{tol_rel_gap} \) tolerance for relative gap between primal and dual problems

Details

This function has been adapted (with only cosmetic changes) from the R package SLOPE version 0.1.3, due to this function being deprecated and defunct in SLOPE versions which are newer than 0.1.3.

Value

An object of class SLOPE_solver.result. This object is a list containing at least the following components:

- \( x \) solution vector \( x \)
- \( \text{optimal} \) logical: whether the solution is optimal
- \( \text{iter} \) number of iterations
Index

admmSolverGroupSLOPE, 2
coef.grpSLOPE, 4
getGroupID, 5, 11
grpSLOPE, 6
lambdaGroupSLOPE, 6, 7, 8
predict.grpSLOPE, 9
prox_sorted_L1, 3, 6, 11, 12, 13
proxGroupSortedL1, 10
proximalGradientSolverGroupSLOPE, 6, 11
sigma, 14
SLOPE_solver, 15