Package: quadrupen (via r-universe)

October 24, 2024

Type Package

Title Sparsity by Worst-Case Quadratic Penalties

Version 0.2-12

Date 2024-06-25

Description Fits classical sparse regression models with efficient active set algorithms by solving quadratic problems as described by Grandvalet, Chiquet and Ambroise (2017) <doi:10.48550/arXiv.1210.2077>. Also provides a few methods for model selection purpose (cross-validation, stability selection).

License GPL (>= 3)

Depends Rcpp, ggplot2, Matrix

Imports reshape2, methods, scales, grid, parallel

Suggests testthat, spelling, lars, elasticnet, glmnet

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Maintainer Julien Chiquet <julien.chiquet@inrae.fr>

URL https://github.com/jchiquet/quadrupenCRAN

BugReports https://github.com/jchiquet/quadrupenCRAN/issues

Encoding UTF-8

RoxygenNote 7.3.1

Language en-US

Author Julien Chiquet [aut, cre] (<https://orcid.org/0000-0002-3629-3429>)

Repository CRAN

Date/Publication 2024-06-25 12:10:02 UTC

Contents

`bounded.reg` *Fit a linear model with infinity-norm plus ridge-like regularization*

Description

Adjust a linear model penalized by a mixture of a (possibly weighted) ℓ_∞-norm (bounding the magnitude of the parameters) and a (possibly structured) ℓ_2-norm (ridge-like). The solution path is computed at a grid of values for the infinity-penalty, fixing the amount of ℓ_2 regularization. See details for the criterion optimized.

Usage

```
bounded.reg(
  x,
  y,
  lambda1 = NULL,
  lambda2 = 0.01,
  penscale = rep(1, p),
  struct = NULL,
  intercept = TRUE,
  normalize = TRUE,
  naive = FALSE,
  nlambda1 = ifelse(is.null(lambda1), 100, length(lambda1)),
  min.ratio = ifelse(n <= p, 0.01, 0.001),
  max.feat = ifelse(lambda2 < 0.01, min(n, p), min(4 * n, p)),
  control = list(),
  checkargs = TRUE
)
```

Arguments

x	matrix of features, possibly sparsely encoded (experimental). Do NOT include intercept. When normalized os TRUE, coefficients will then be rescaled to the original scale.
y	response vector.
lambda1	sequence of decreasing ℓ_∞ penalty levels. If NULL (the default), a vector is generated with nlambda1 entries, starting from a guessed level lambda1.max where only the intercept is included, then shrunken to min.ratio*lambda1.max.
lambda2	real scalar; tunes the ℓ_2-penalty in the bounded regression. Default is 0.01. Set to 0 to regularize only by the infinity norm (be careful regarding numerical stability in that case, particularly in the high dimensional setting).
penscale	vector with real positive values that weight the infinity norm of each feature. Default set all weights to 1. See details below.
struct	matrix structuring the coefficients. Must be at least positive semidefinite (this is checked internally if the checkarg argument is TRUE). The default uses the identity matrix. See details below.
intercept	logical; indicates if an intercept should be included in the model. Default is TRUE.
normalize	logical; indicates if variables should be normalized to have unit L2 norm before fitting. Default is TRUE.
naive	logical; Compute either 'naive' of 'classic' bounded regression: mimicking the Elastic-net, the vector of parameters is rescaled by a coefficient (1+lambda2) when naive equals FALSE. No rescaling otherwise. Default is FALSE.
nlambda1	integer that indicates the number of values to put in the lambda1 vector. Ignored if lambda1 is provided.
min.ratio	minimal value of infinity-part of the penalty that will be tried, as a fraction of the maximal lambda1 value. A too small value might lead to unstability at the end of the solution path corresponding to small lambda1. The default value tries to avoid this, adapting to the '$n < p$' context. Ignored if lambda1 is provided.
max.feat	integer; limits the number of features ever to enter the model: in our implementation of the bounded regression, it corresponds to the variables which have left the boundary along the path. The algorithm stops if this number is exceeded and lambda1 is cut at the corresponding level. Default is min(nrow(x),ncol(x)) for small lambda2 (<0.01) and min(4*nrow(x),ncol(x)) otherwise. Use with care, as it considerably changes the computation time.
control	list of argument controlling low level options of the algorithm –use with care and at your own risk– :

verbose: integer; activate verbose mode –this one is not too much risky!– set to 0 for no output; 1 for warnings only, and 2 for tracing the whole progression. Default is 1. Automatically set to 0 when the method is embedded within cross-validation or stability selection.

timer: logical; use to record the timing of the algorithm. Default is FALSE.

max.iter: the maximal number of iteration used to solve the problem for a given value of lambda1. Default is 500.

`method`: a string for the underlying solver used. Either `"quadra"` or `"fista"` are available for bounded regression. Default is `"quadra"`.

`threshold`: a threshold for convergence. The algorithm stops when the optimality conditions are fulfill up to this threshold. Default is `1e-7` for `"quadra"` and `1e-2` for `"fista"`.

`bulletproof`: logical; indicates if the bulletproof mode should be used while running the `"quadra"` method. Default is `TRUE`. See details below.

`checkargs` logical; should arguments be checked to (hopefully) avoid internal crashes? Default is `TRUE`. Automatically set to `FALSE` when calls are made from cross-validation or stability selection procedures.

Details

The optimized criterion is

$$\hat{\beta}_{\lambda_1,\lambda_1} = \arg\min_{\beta} \frac{1}{2}(y - X\beta)^T(y - X\beta) + \lambda_1\|D\beta\|_\infty + \frac{\lambda_2}{2}\beta^T S\beta,$$

where D is a diagonal matrix, whose diagonal terms are provided as a vector by the `penscale` argument. The ℓ_2 structuring matrix S is provided via the `struct` argument, a positive semidefinite matrix (possibly of class `Matrix`). Note that the quadratic algorithm for the bounded regression may become unstable along the path because of singularity of the underlying problem, e.g. when there are too much correlation or when the size of the problem is close to or smaller than the sample size. In such cases, it might be a good idea to switch to the proximal solver, slower yet more robust. This is the strategy adopted by the `'bulletproof'` mode, that will send a warning while switching the method to `'fista'` and keep on optimizing on the remainder of the path. When `bulletproof` is set to `FALSE`, the algorithm stops at an early stage of the path of solutions. Hence, users should be careful when manipulating the resulting `'quadrupen'` object, as it will not have the size expected regarding the dimension of the `lambda1` argument.

Singularity of the system can also be avoided with a larger ℓ_2-regularization, via `lambda2`, or a "not-too-small" ℓ_∞ regularization, via a larger `'min.ratio'` argument.

Value

an object with class `quadrupen`, see the documentation page `quadrupen` for details.

See Also

See also `quadrupen`, `plot,quadrupen-method` and `crossval`.

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
cor <- 0.75
Soo <- toeplitz(cor^(0:(25-1))) ## Toeplitz correlation for irrelevant variables
Sww  <- matrix(cor,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo)
```

```
diag(Sigma) <- 1
n <- 50
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

## Infinity norm without/with an additional l2 regularization term
## and with structuring prior
labels <- rep("irrelevant", length(beta))
labels[beta != 0] <- "relevant"
plot(bounded.reg(x,y,lambda2=0) , label=labels) ## a mess
plot(bounded.reg(x,y,lambda2=10), label=labels) ## good guys are at the boundaries
```

`crossval` *Cross-validation function for quadrupen fitting methods.*

Description

Function that computes K-fold (double) cross-validated error of a `quadrupen` fit. If no `lambda2` is provided, simple cross validation on the `lambda1` parameter is performed. If a vector `lambda2` is passed as an argument, double cross-validation is performed.

Usage

```
crossval(
  x,
  y,
  penalty = c("elastic.net", "bounded.reg"),
  K = 10,
  folds = split(sample(1:nrow(x)), rep(1:K, length = nrow(x))),
  lambda2 = 0.01,
  verbose = TRUE,
  mc.cores = 2,
  ...
)
```

Arguments

`x`	matrix of features, possibly sparsely encoded (experimental). Do NOT include intercept.
`y`	response vector.
`penalty`	a string for the fitting procedure used for cross-validation. Either `"elastic.net"` or `"bounded.reg"`, at the moment. Default is `elastic.net`.
`K`	integer indicating the number of folds. Default is 10.
`folds`	list of `K` vectors that describes the folds to use for the cross-validation. By default, the folds are randomly sampled with the specified K. The same folds are used for each values of `lambda2`.

lambda2	tunes the ℓ_2-penalty (ridge-like) of the fit. If none is provided, the default scalar value of the corresponding fitting method is used and a simple CV is performed. If a vector of values is given, double cross-validation is performed (both on lambda1 and lambda2, using the same folds for each lambda2).
verbose	logical; indicates if the progression (the current lambda2) should be displayed. Default is TRUE.
mc.cores	the number of cores to use. The default uses 2 cores.
...	additional parameters to overwrite the defaults of the fitting procedure identified by the 'penalty' argument. See the corresponding documentation (elastic.net or bounded.reg).

Value

An object of class "cvpen" for which a plot method is available.

Note

If the user runs the fitting method with option 'bulletproof' set to FALSE, the algorithm may stop at an early stage of the path. Early stops are handled internally, in order to provide results on the same grid of penalty tuned by λ_1. This is done by means of NA values, so as mean and standard error are consistently evaluated. If, while cross-validating, the procedure experiences too many early stoppings, a warning is sent to the user, in which case you should reconsider the grid of lambda1 used for the cross-validation. If bulletproof is TRUE (the default), there is nothing to worry about, except a possible slow down when any switching to the proximal algorithm is required.

See Also

quadrupen, plot,cvpen-method and cvpen.

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
cor  <- 0.75
Soo  <- toeplitz(cor^(0:(25-1))) ## Toeplitz correlation for irrelevant variable
Sww  <- matrix(cor,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo) + 0.1
diag(Sigma) <- 1
n <- 100
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

## Use fewer lambda1 values by overwritting the default parameters
## and cross-validate over the sequences lambda1 and lambda2

cv.double <- crossval(x,y, lambda2=10^seq(2,-2,len=50), nlambda1=50)

## Rerun simple cross-validation with the appropriate lambda2
cv.10K <- crossval(x,y, lambda2=0.2)
```

```
## Try leave one out also
cv.loo <- crossval(x,y, K=n, lambda2=0.2)


plot(cv.double)

plot(cv.10K)
plot(cv.loo)

## Performance for selection purpose
beta.min.10K <- slot(cv.10K, "beta.min")
beta.min.loo <- slot(cv.loo, "beta.min")

cat("\nFalse positives with the minimal 10-CV choice: ", sum(sign(beta) != sign(beta.min.10K)))
cat("\nFalse positives with the minimal LOO-CV choice: ", sum(sign(beta) != sign(beta.min.loo)))
```

`cvpen-class` *Class "cvpen"*

Description

Class of object returned by a cross-validation performed through the `crossval` method.

Slots

`lambda1`: vector of λ_1 (ℓ_1 or ℓ_∞ penalty levels) for which each cross-validation has been performed.

`lambda2`: vector (or scalar) of ℓ_2-penalty levels for which each cross-validation has been performed.

`lambda1.min`: level of λ_1 that minimizes the error estimated by cross-validation.

`lambda1.1se`: largest level of λ_1 such as the cross-validated error is within 1 standard error of the minimum.

`lambda2.min`: level of λ_2 that minimizes the error estimated by cross-validation.

`cv.error`: a data frame containing the mean cross-validated error and its associated standard error for each values of `lambda1` and `lambda2`.

`folds`: list of `K` vectors indicating the folds used for cross-validation.

`beta.min`: the vector of parameters obtained by fitting the problem on the full data set `x` and `y` with `lambda1.min` and `lambda2.min` penalties.

`beta.1se`: the vector of parameters obtained by fitting the problem on the full data set `x` and `y` with `lambda1.1se` and `lambda2.min` penalties.

The specific `plot,cvpen-method` method is documented.

See Also

See also `plot,cvpen-method` and `crossval`.

`elastic.net` *Fit a linear model with elastic-net regularization*

Description

Adjust a linear model with elastic-net regularization, mixing a (possibly weighted) ℓ_1-norm (LASSO) and a (possibly structured) ℓ_2-norm (ridge-like). The solution path is computed at a grid of values for the ℓ_1-penalty, fixing the amount of ℓ_2 regularization. See details for the criterion optimized.

Usage

```
elastic.net(
  x,
  y,
  lambda1 = NULL,
  lambda2 = 0.01,
  penscale = rep(1, p),
  struct = NULL,
  intercept = TRUE,
  normalize = TRUE,
  naive = FALSE,
  nlambda1 = ifelse(is.null(lambda1), 100, length(lambda1)),
  min.ratio = ifelse(n <= p, 0.01, 1e-04),
  max.feat = ifelse(lambda2 < 0.01, min(n, p), min(4 * n, p)),
  beta0 = NULL,
  control = list(),
  checkargs = TRUE
)
```

Arguments

`x`	matrix of features, possibly sparsely encoded (experimental). Do NOT include intercept. When normalized os `TRUE`, coefficients will then be rescaled to the original scale.
`y`	response vector.
`lambda1`	sequence of decreasing ℓ_1-penalty levels. If `NULL` (the default), a vector is generated with `nlambda1` entries, starting from a guessed level `lambda1.max` where only the intercept is included, then shrunken to `min.ratio*lambda1.max`.
`lambda2`	real scalar; tunes the ℓ_2 penalty in the Elastic-net. Default is 0.01. Set to 0 to recover the Lasso.
`penscale`	vector with real positive values that weight the ℓ_1-penalty of each feature. Default set all weights to 1.
`struct`	matrix structuring the coefficients (preferably sparse). Must be at least positive semidefinite (this is checked internally if the `checkarg` argument is `TRUE`). The default uses the identity matrix. See details below.

`intercept` logical; indicates if an intercept should be included in the model. Default is `TRUE`.

`normalize` logical; indicates if variables should be normalized to have unit L2 norm before fitting. Default is `TRUE`.

`naive` logical; Compute either 'naive' of classic elastic-net as defined in Zou and Hastie (2006): the vector of parameters is rescaled by a coefficient `(1+lambda2)` when `naive` equals `FALSE`. No rescaling otherwise. Default is `FALSE`.

`nlambda1` integer that indicates the number of values to put in the `lambda1` vector. Ignored if `lambda1` is provided.

`min.ratio` minimal value of ℓ_1-part of the penalty that will be tried, as a fraction of the maximal `lambda1` value. A too small value might lead to unstability at the end of the solution path corresponding to small `lambda1` combined with $\lambda_2 = 0$. The default value tries to avoid this, adapting to the '$n < p$' context. Ignored if `lambda1` is provided.

`max.feat` integer; limits the number of features ever to enter the model; i.e., non-zero coefficients for the Elastic-net: the algorithm stops if this number is exceeded and `lambda1` is cut at the corresponding level. Default is `min(nrow(x),ncol(x))` for small `lambda2` (<0.01) and `min(4*nrow(x),ncol(x))` otherwise. Use with care, as it considerably changes the computation time.

`beta0` a starting point for the vector of parameter. When `NULL` (the default), will be initialized at zero. May save time in some situation.

`control` list of argument controlling low level options of the algorithm –use with care and at your own risk– :

- `verbose:` integer; activate verbose mode –this one is not too much risky!– set to `0` for no output; `1` for warnings only, and `2` for tracing the whole progression. Default is `1`. Automatically set to `0` when the method is embedded within cross-validation or stability selection.
- `timer:` logical; use to record the timing of the algorithm. Default is `FALSE`.
- `max.iter:` the maximal number of iteration used to solve the problem for a given value of lambda1. Default is 500.
- `method:` a string for the underlying solver used. Either `"quadra"`, `"pathwise"` or `"fista"`. Default is `"quadra"`.
- `threshold:` a threshold for convergence. The algorithm stops when the optimality conditions are fulfill up to this threshold. Default is `1e-7` for `"quadra"` and `1e-2` for the first order methods.
- `monitor:` indicates if a monitoring of the convergence should be recorded, by computing a lower bound between the current solution and the optimum: when `'0'` (the default), no monitoring is provided; when `'1'`, the bound derived in Grandvalet et al. is computed; when `'>1'`, the Fenchel duality gap is computed along the algorithm.

`checkargs` logical; should arguments be checked to (hopefully) avoid internal crashes? Default is `TRUE`. Automatically set to `FALSE` when calls are made from cross-validation or stability selection procedures.

Details

The optimized criterion is the following:

$$\hat{\beta}_{\lambda_1,\lambda_1} = \arg\min_{\beta} \frac{1}{2}(y - X\beta)^T(y - X\beta) + \lambda_1\|D\beta\|_1 + \frac{\lambda_2}{2}\beta^T S\beta,$$

where D is a diagonal matrix, whose diagonal terms are provided as a vector by the `penscale` argument. The ℓ_2 structuring matrix S is provided via the `struct` argument, a positive semidefinite matrix (possibly of class `Matrix`).

Value

an object with class quadrupen, see the documentation page quadrupen for details.

See Also

See also quadrupen, plot,quadrupen-method and crossval.

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
cor <- 0.75
Soo <- toeplitz(cor^(0:(25-1))) ## Toeplitz correlation for irrelevant variables
Sww  <- matrix(cor,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo)
diag(Sigma) <- 1
n <- 50
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

labels <- rep("irrelevant", length(beta))
labels[beta != 0] <- "relevant"
## Comparing the solution path of the LASSO and the Elastic-net
plot(elastic.net(x,y,lambda2=0), label=labels) ## a mess
plot(elastic.net(x,y,lambda2=10), label=labels) ## a lot better
```

---

plot,cvpen-method *Plot method for cross validated error of a* quadrupen *model*

Description

Produce a plot of the cross validated error of a quadrupen model.

Usage

```
\S4method{plot}{cvpen}(x, y, log.scale=TRUE, reverse=FALSE,
plot=TRUE, main = "Cross-validation error", ...)
```

Arguments

x	output of a crossval run (must be of class cvpen).
y	used for S4 compatibility.
log.scale	logical; indicates if a log-scale should be used when xvar="lambda". Ignored for 2D cross-validation plot.
reverse	logical; should the X-axis by reversed when xvar=lambda? Default is FALSE. Ignored for 2D cross-validation plot.
plot	logical; indicates if the graph should be plotted. Default is TRUE.
main	the main title, with a hopefully appropriate default definition.
...	used for S4 compatibility.

Value

a **ggplot2** object which can be plotted via the print method.

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
cor  <- 0.75
Soo  <- toeplitz(cor^(0:(25-1))) ## Toeplitz correlation for irrelevant variables
Sww  <- matrix(cor,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo) + 0.1
diag(Sigma) <- 1
n <- 100
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

## Use fewer lambda1 values by overwritting the default parameters
## and cross-validate over the sequences lambda1 and lambda2

cv.double <- crossval(x,y, lambda2=10^seq(2,-2,len=50), nlambda1=50)

## Rerun simple cross-validation with the appropriate lambda2
cv.10K <- crossval(x,y, lambda2=.2)
## Try leave one out also
cv.loo <- crossval(x,y, K=n, lambda2=0.2)


plot(cv.double)

plot(cv.10K)
plot(cv.loo)

## Performance for selection purpose
beta.min.10K <- slot(cv.10K, "beta.min")
beta.min.loo <- slot(cv.loo, "beta.min")
```

```
cat("\nFalse positives with the minimal 10-CV choice: ", sum(sign(beta) != sign(beta.min.10K)))
cat("\nFalse positives with the minimal LOO-CV choice: ", sum(sign(beta) != sign(beta.min.loo)))
```

`plot,quadrupen-method` *Plot method for a quadrupen object*

Description

Produce a plot of the solution path of a quadrupen fit.

Usage

```
\S4method{plot}{quadrupen}(x, y, xvar = "lambda",
        main = paste(slot(x, "penalty")," path", sep=""),
        log.scale = TRUE, standardize=TRUE, reverse=FALSE,
        labels = NULL, plot = TRUE, ...)
```

Arguments

x	output of a fitting procedure of the **quadrupen** package (elastic.net or bounded.reg for the moment). Must be of class quadrupen.
y	used for S4 compatibility.
xvar	variable to plot on the X-axis: either "lambda" (λ_1 penalty level) or "fraction" (ℓ_1-norm of the coefficients). Default is set to "lambda".
main	the main title. Default is set to the model name followed by what is on the Y-axis.
log.scale	logical; indicates if a log-scale should be used when xvar="lambda". Default is TRUE.
standardize	logical; standardize the coefficients before plotting (with the norm of the predictor). Default is TRUE.
reverse	logical; should the X-axis be reversed when xvar="lambda"? Default is FALSE.
labels	vector indicating the names associated to the plotted variables. When specified, a legend is drawn in order to identify each variable. Only relevant when the number of predictor is small. Remind that the intercept does not count. Default is NULL.
plot	logical; indicates if the graph should be plotted on call. Default is TRUE.
...	Not used

Value

a **ggplot2** object which can be plotted via the print method.

See Also

quadrupen.

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
cor <- 0.75
Soo <- toeplitz(cor^(0:(25-1))) ## Toeplitz correlation for irrelevant variables
Sww  <- matrix(cor,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo)
diag(Sigma) <- 1
n <- 50
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

## Plot the Lasso path
plot(elastic.net(x,y, lambda2=0), main="Lasso solution path")
## Plot the Elastic-net path
plot(elastic.net(x,y, lambda2=10), xvar = "lambda")
## Plot the Elastic-net path (fraction on X-axis, unstandardized coefficient)
plot(elastic.net(x,y, lambda2=10), standardize=FALSE, xvar="fraction")
## Plot the Bounded regression path (fraction on X-axis)
plot(bounded.reg(x,y, lambda2=10), xvar="fraction")
```

---

plot,stability.path-method

Plot method for stability.path.

Description

Produce a plot of the stability path obtained by stability selection.

Usage

```
\S4method{plot}{stability.path}(x, y, xvar = "lambda", annot=TRUE,
      main = paste("Stability path for ", slot(x, "penalty")," regularizer", sep=""),
         log.scale = TRUE,  labels = rep("unknown status",p), plot = TRUE,
      sel.mode = c("rank","PFER"), cutoff=0.75, PFER=2, nvar=floor(n/log(p)), ...)
```

Arguments

x	output of a stability run (must be of class stability.path).
y	used for S4 compatibility.
xvar	variable to plot on the X-axis: either "lambda" (first penalty level) or "fraction" (fraction of the penalty level applied tune by λ_1). Default is "lambda".
annot	logical; should annotation be made on the graph regarding controlled PFER (only relevant when sel.mode equals 'PFER')? Default is TRUE.

main	main title. If none given, a somewhat appropriate title is automatically generated.
log.scale	logical; indicates if a log-scale should be used when xvar="lambda". Default is TRUE.
labels	an optional vector of labels for each variable in the path (e.g., 'relevant'/'irrelevant'). See examples.
plot	logical; indicates if the graph should be plotted. Default is TRUE. If FALSE, only the **ggplot2** object is sent back.
sel.mode	a character string, either 'rank' or 'PFER'. In the first case, the selection is based on the rank of total probabilities by variables along the path: the first nvar variables are selected (see below). In the second case, the PFER control is used as described in Meinshausen and Buhlmann's paper. Default is 'rank'.
cutoff	value of the cutoff probability (only relevant when sel.mode equals 'PFER').
PFER	value of the per-family error rate to control (only relevant when sel.mode equals 'PFER').
nvar	number of variables selected (only relevant when sel.mode equals 'rank'. Default is floor(n/log(p)).
...	used for S4 compatibility.

Value

a list with a **ggplot2** object which can be plotted via the print method, and a vector of selected variables corresponding to method of choice ('rank' or 'PFER')

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
Soo  <- matrix(0.75,25,25) ## bloc correlation between zero variables
Sww  <- matrix(0.75,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo) + 0.2
diag(Sigma) <- 1
n <- 100
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

## Build a vector of label for true nonzeros
labels <- rep("irrelevant", length(beta))
labels[beta != 0] <- c("relevant")
labels <- factor(labels, ordered=TRUE, levels=c("relevant","irrelevant"))

## Call to stability selection function, 200 subsampling
stab <- stability(x,y, subsamples=200, lambda2=1, min.ratio=1e-2)

## Build the plot an recover the selected variable
plot(stab, labels=labels)
plot(stab, xvar="fraction", labels=labels, sel.mode="PFER", cutoff=0.75, PFER=2)
```

quadrupen-class *Class "quadrupen"*

Description

Class of object returned by any fitting function of the **quadrupen** package (`elastic.net` or `bounded.reg`).

Slots

`coefficients`: Matrix (class `"dgCMatrix"`) of coefficients with respect to the original input. The number of rows corresponds the length of `lambda1`.

`active.set`: Matrix (class `"dgCMatrix"`, generally sparse) indicating the 'active' variables, in the sense that they activate the constraints. For the `elastic.net`, it corresponds to the nonzero variables; for the `bounded.reg` function, it is the set of variables reaching the boundary along the path of solutions.

`intercept`: logical; indicates if an intercept has been included to the model.

`mu`: A vector (class `"numeric"`) containing the successive values of the (unpenalized) intercept. Equals to zero if `intercept` has been set to `FALSE`.

`meanx`: Vector (class `"numeric"`) containing the column means of the predictor matrix.

`normx`: Vector (class `"numeric"`) containing the square root of the sum of squares of each column of the design matrix.

`penscale`: Vector `"numeric"` with real positive values that have been used to weight the penalty tuned by λ_1.

`penalty`: Object of class `"character"` indicating the method used (`"elastic-net"` or `"bounded regression"`).

`naive`: logical; was the `naive` mode on?

`lambda1`: Vector (class `"numeric"`) of penalty levels (either ℓ_1 or ℓ_∞) for which the model has eventually been fitted.

`lambda2`: Scalar (class `"numeric"`) for the amount of ℓ_2 (ridge-like) penalty.

`struct`: Object of class `"Matrix"` used to structure the coefficients in the ℓ_2 penalty.

`control`: Object of class `"list"` with low level options used for optimization.

`monitoring`: List (class `"list"`) which contains various indicators dealing with the optimization process.

`residuals`: Matrix of residuals, each column corresponding to a value of `lambda1`.

`r.squared`: Vector (class `"numeric"`) given the coefficient of determination as a function of lambda1.

`fitted`: Matrix of fitted values, each column corresponding to a value of `lambda1`.

Methods

This class comes with the usual `predict(object, newx, ...)`, `fitted(object, ...)`, `residuals(object, ...)`, `print(object, ...)`, `show(object)` and `deviance(object, ...)` generic (undocumented) methods.

A specific plotting method is available and documented (`plot,quadrupen-method`).

See Also

See also plot,quadrupen-method.

---

stability *Stability selection for a quadrupen fit.*

Description

Compute the stability path of a (possibly randomized) fitting procedure as introduced by Meinshausen and Buhlmann (2010).

Usage

```
stability(
  x,
  y,
  penalty = c("elastic.net", "bounded.reg"),
  subsamples = 100,
  sample.size = floor(n/2),
  randomize = TRUE,
  weakness = 0.5,
  verbose = TRUE,
 folds = replicate(subsamples, sample(1:nrow(x), sample.size), simplify = FALSE),
  mc.cores = 2,
  ...
)
```

Arguments

x	matrix of features, possibly sparsely encoded (experimental). Do NOT include intercept.
y	response vector.
penalty	a string for the fitting procedure used for cross-validation. Either elastic.net or "bounded.reg".
subsamples	integer indicating the number of subsamplings used to estimate the selection probabilities. Default is 100.
sample.size	integer indicating the size of each subsamples. Default is floor(n/2).
randomize	Should a randomized version of the fitting procedure by used? Default is TRUE. See details below.
weakness	Coefficient used for randomizing. Default is 0.5. Ignored when randomized is FALSE. See details below.
verbose	logical; indicates if the progression should be displayed. Default is TRUE.
folds	list with subsamples entries with vectors describing the folds to use for the stability procedure. By default, the folds are randomly sampled with the specified subsamples argument.

mc.cores	the number of cores to use. The default uses 2 cores.
...	additional parameters to overwrite the defaults of the fitting procedure. See the corresponding documentation (elastic.net or bounded.reg)

Value

An object of class stability.path.

Note

When randomized = TRUE, the penscale argument that weights the penalty tuned by λ_1 is perturbed (divided) for each subsample by a random variable uniformly distributed on $[\alpha, 1]$, where α is the weakness parameter.

If the user runs the fitting method with option 'bulletproof' set to FALSE, the algorithm may stop at an early stage of the path. Early stops of the underlying fitting function are handled internally, in the following way: we chose to simply skip the results associated with such runs, in order not to bias the stability selection procedure. If it occurs too often, a warning is sent to the user, in which case you should reconsider the grid of lambda1 for stability selection. If bulletproof is TRUE (the default), there is nothing to worry about, except a possible slow down when any switching to the proximal algorithm is required.

References

N. Meinshausen and P. Buhlmann (2010). Stability Selection, JRSS(B).

See Also

stability.path and plot,stability.path-method.

Examples

```
## Simulating multivariate Gaussian with blockwise correlation
## and piecewise constant vector of parameters
beta <- rep(c(0,1,0,-1,0), c(25,10,25,10,25))
Soo  <- matrix(0.75,25,25) ## bloc correlation between zero variables
Sww  <- matrix(0.75,10,10) ## bloc correlation between active variables
Sigma <- bdiag(Soo,Sww,Soo,Sww,Soo) + 0.2
diag(Sigma) <- 1
n <- 100
x <- as.matrix(matrix(rnorm(95*n),n,95) %*% chol(Sigma))
y <- 10 + x %*% beta + rnorm(n,0,10)

## Build a vector of label for true nonzeros
labels <- rep("irrelevant", length(beta))
labels[beta != 0] <- c("relevant")
labels <- factor(labels, ordered=TRUE, levels=c("relevant","irrelevant"))

## Call to stability selection function, 200 subsampling
stab <- stability(x,y, subsamples=200, lambda2=1, min.ratio=1e-2)
## Recover the selected variables for a given cutoff
## and per-family error rate, without producing any plot
```

```
stabpath <- plot(stab, cutoff=0.75, PFER=1, plot=FALSE)

cat("\nFalse positives for the randomized Elastic-net with stability selection: ",
     sum(labels[stabpath$selected] != "relevant"))
cat("\nDONE.\n")
```

`stability.path-class` *Class "stability.path"*

Description

Class of object returned by the `stability` function, with methods `print`, `show` and `plot`.

Slots

`probabilities`: a `Matrix` object containing the estimated probabilities of selection along the path of solutions.

`penalty`: Object of class `"character"` indicating the penalizer used.

`naive`: logical indicating whether rescaling of the coefficients has been performed regarding the ℓ_2-penalty.

`lambda1`: a vector with the levels of the first penalty.

`lambda2`: a scalar with the ℓ_2-penalty level.

`folds`: a list that contains the folds used for each subsample.

See Also

See also `plot,stability.path-method`, and `stability`.

Index