Package 'stabs'

October 2, 2014

Title Stability Selection with Error Control

Version 0.1-0

Date 2014-10-02

Description Resampling procedures to assess the stability of selected variables with additional finite sample error control for high-dimensional variable selection procedures such as Lasso or boosting

Depends R (>= 2.14.0), methods, stats, parallel

Suggests TH.data, hdi

Enhances glmnet, lars, mboost (> 2.3-0)

LazyData yes

License GPL-2

URL http://r-forge.r-project.org/projects/stabsel/

Author Benjamin Hofner [aut, cre],Torsten Hothorn [aut]

Maintainer Benjamin Hofner <benjamin.hofner@fau.de>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-10-02 15:43:49

R topics documented:

`Fitting Functions` *Fit Functions for Stability Selection*

Description

Functions that fit a model until q variables are selected and that returns the indices (and names) of the selected variables.

Usage

```
## package lars:
lars.lasso(x, y, q, ...)
lars.stepwise(x, y, q, ...)

## package glmnet:
glmnet.lasso(x, y, q, ...)
```

Arguments

`x`	a matrix containing the predictors or an object of class `"mboost"`.
`y`	a vector or matrix containing the outcome.
`q`	number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
`...`	additional arguments to the underlying fitting function.

Details

All fitting functions are named after the package and the type of model that is fitted: `package_name.model`, e.g., `glmnet.lasso` stands for a lasso model that is fitted using the package **glmnet**.

Value

A named list with elements

`selected`	logical. A vector that indicates which variable was selected.
`path`	logical. A matrix that indicates which variabkle was selected in which step. Each row represents one variable, the columns represent the steps.

Examples

```
data("bodyfat", package = "TH.data")
## selected variables
lars.lasso(bodyfat[, -2], bodyfat[,2], q = 3)$selected
lars.stepwise(bodyfat[, -2], bodyfat[,2], q = 3)$selected
glmnet.lasso(bodyfat[, -2], bodyfat[,2], q = 3)$selected
```

plot.stabsel *Plot and Print Methods for Stability Selection*

Description

Display results of stability selection.

Usage

```
## S3 method for class 'stabsel'
plot(x, main = deparse(x$call), type = c("maxsel", "paths"),
     xlab = NULL, ylab = NULL, col = NULL, ymargin = 10, np = sum(x$max > 0),
     labels = NULL, ...)
## S3 method for class 'stabsel'
print(x, decreasing = FALSE, print.all = TRUE, ...)
```

Arguments

x	object of class stabsel.
main	main title for the plot.
type	plot type; either stability paths ("paths") or a plot of the maximum selection frequency ("maxsel").
xlab, ylab	labels for the x- and y-axis of the plot. Per default, sensible labels are used depending on the type of the plot.
col	a vector of colors; Typically, one can specify a single color or one color for each variable. Per default, colors depend on the maximal selection frequency of the variable and range from grey to red.
ymargin	(temporarily) specifies the y margin of of the plot in lines (see argument "mar" of function par). This only affects the right margin for type = "paths" and the left margin for type = "maxsel". Explicit user specified margins are kept and are not overwritten.
np	number of variables to plot for the maximum selection frequency plot (type = "maxsel"); the first np variables with highest selection frequency are plotted.
labels	variable labels for the plot; one label per variable / effect must be specified. Per default, the names of x$max are used.
decreasing	logical. Should the selection frequencies be printed in descending order (TRUE) or in ascending order (FALSE)?
print.all	logical. Should all selection frequencies be displayed or only those that are greater than zero?
...	additional arguments to plot and print functions.

Details

This function implements the stability selection procedure by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013).

Two of the three arguments `cutoff`, `q` and `PFER` *must* be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where V is the number of false positives, is bounded by the argument `PFER`.

As controlling the PFER is more conservative as controlling the family-wise error rate (FWER), the procedure also controlls the FWER, i.e., the probability of selecting at least one non-influential variable (or model component) is less than `PFER`.

Value

An object of class `stabsel` with a special `print` method. The object has the following elements:

`phat`	selection probabilities.
`selected`	elements with maximal selection probability greater `cutoff`.
`max`	maximum of selection probabilities.
`cutoff`	cutoff used.
`q`	average number of selected variables used.
`PFER`	per-family error rate.
`sampling.type`	the sampling type used for stability selection.
`assumption`	the assumptions made on the selection probabilities.
`call`	the call.

References

N. Meinshausen and P. Buehlmann (2010), Stability selection. *Journal of the Royal Statistical Society, Series B*, **72**, 417–473.

R.D. Shah and R.J. Samworth (2013), Variable selection with error control: another look at stability selection. *Journal of the Royal Statistical Society, Series B*, **75**, 55–80.

See Also

`stabsel`

Examples

```
data("bodyfat", package = "TH.data")
## set seed
set.seed(1234)

######################################################################
### using stability selection with Lasso methods:

if (require("lars")) {
    (stab.lasso <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                           fitfun = lars.lasso, cutoff = 0.75,
```

```
                              PFER = 1))
    par(mfrow = c(2, 1))
    plot(stab.lasso, ymargin = 6)
    opar <- par(mai = par("mai") * c(1, 1, 1, 2.7))
    plot(stab.lasso, type = "paths")
}
```

---

stabsel *Stability Selection*

Description

Selection of influential variables or model components with error control.

Usage

```
## generic stability selection funcion
stabsel(x, ...)

## a method to fit models with stability selection
## S3 method for class 'matrix'
stabsel(x, y, fitfun = lars.lasso,
        args.fitfun = list(), cutoff, q, PFER,
        folds = subsample(rep(1, nrow(x)), B = B),
        B = ifelse(sampling.type == "MB", 100, 50),
        assumption = c("unimodal", "r-concave", "none"),
        sampling.type = c("SS", "MB"),
        papply = mclapply, verbose = TRUE, FWER, eval = TRUE, ...)

## essentially a wrapper for data.frames (see details)
## S3 method for class 'data.frame'
stabsel(x,  y, intercept = FALSE, ...)
```

Arguments

x	a matrix or a data.frame containing the predictors.
y	a vector or matrix containing the outcome.
intercept	logical. If x is a data.frame, this argument determines if the resulting model matrix should contain a separate intercept or not.
fitfun	a function that takes the arguments x, y as above, and additionally the number of variables to include in each model q. The function then needs to fit the model and to return a logical vector that indicates which variable was selected (among the q selected variables).
args.fitfun	a named list containing additional arguments that are passed to the fitting function; see also argument args in do.call.

cutoff	cutoff between 0.5 and 1. Preferably a value between 0.6 and 0.9 should be used.
q	number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
PFER	upper bound for the per-family error rate. This specifies the amount of falsely selected base-learners, which is tolerated. See details.
folds	a weight matrix with number of rows equal to the number of observations, see subsample. Usually one should not change the default here as subsampling with a fraction of $1/2$ is needed for the error bounds to hold. One usage scenario where specifying the folds by hand might be the case when one has dependent data (e.g. clusters) and thus wants to draw clusters (i.e., multiple rows together) not individuals.
assumption	Defines the type of assumptions on the distributions of the selection probabilities and simultaneous selection probabilities. Only applicable for sampling.type = "SS". For sampling.type = "MB" we always use code"none".
sampling.type	use sampling scheme of of Shah & Samworth (2013), i.e., with complementarty pairs (sampling.type = "SS"), or the original sampling scheme of Meinshausen & Buehlmann (2010).
B	number of subsampling replicates. Per default, we use 50 complementary pairs for the error bounds of Shah & Samworth (2013) and 100 for the error bound derived in Meinshausen & Buehlmann (2010). As we use B complementray pairs in the former case this leads to $2B$ subsamples.
papply	(parallel) apply function, defaults to mclapply. Alternatively, parLapply can be used. In the latter case, usually more setup is needed (see example of cvrisk for some details).
verbose	logical (default: TRUE) that determines wether warnings should be issued.
FWER	deprecated. Only for compatibility with older versions, use PFER instead.
eval	logical. Determines whether stability selection is evaluated (eval = TRUE; default) or if only the parameter combination is returned.
...	additional arguments to parallel apply methods such as mclapply.

Details

This function implements the stability selection procedure by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013). The error bounds are implemented in the function stabsel_parameters. Two of the three arguments cutoff, q and PFER *must* be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where V is the number of false positives, is bounded by the argument PFER.

As controlling the PFER is more conservative as controlling the family-wise error rate (FWER), the procedure also controlls the FWER, i.e., the probability of selecting at least one non-influential variable (or model component) is less than PFER.

Predefined fitfuns functions exist but more can be easily implemented. Note that stepwise regression methods are usually not advised as they tend to be relatively unstable. See example below.

The function stabsel for data.frames is essentially just a wrapper to the matrix function with the same argments. The only difference is that in a pre-processing step, the data set is converted to a

model matrix using the function model.matrix. The additional argument intercept determines if an explicit intercept should be added to the model matrix. This is often not neccessary but depends on the fitfun.

Value

An object of class stabsel with a special print method. The object has the following elements:

phat	selection probabilities.
selected	elements with maximal selection probability greater cutoff.
max	maximum of selection probabilities.
cutoff	cutoff used.
q	average number of selected variables used.
PFER	per-family error rate.
sampling.type	the sampling type used for stability selection.
assumption	the assumptions made on the selection probabilities.
call	the call.

References

N. Meinshausen and P. Buehlmann (2010), Stability selection. *Journal of the Royal Statistical Society, Series B*, **72**, 417–473.

R.D. Shah and R.J. Samworth (2013), Variable selection with error control: another look at stability selection. *Journal of the Royal Statistical Society, Series B*, **75**, 55–80.

See Also

stabsel_parameters for the computation of error bounds, fitfun for available fitting functions and plot.stabsel for available plot functions

Examples

```
## make data set available
data("bodyfat", package = "TH.data")
## set seed
set.seed(1234)

######################################################################
### using stability selection with Lasso methods:

if (require("lars")) {
    (stab.lasso <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                           fitfun = lars.lasso, cutoff = 0.75,
                           PFER = 1))
    (stab.stepwise <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                              fitfun = lars.stepwise, cutoff = 0.75,
                              PFER = 1))
    par(mfrow = c(2, 1))
```

```
    plot(stab.lasso, main = "Lasso")
    plot(stab.stepwise, main = "Stepwise Selection")
    ## --> stepwise selection seems to be quite unstable even in this low
    ##     dimensional example!
}

######################################################################
### using stability selection directly on computed boosting models
### from mboost

if (require("mboost")) {
    ### low-dimensional example
    mod <- glmboost(DEXfat ~ ., data = bodyfat)

    ## compute cutoff ahead of running stabsel to see if it is a sensible
    ## parameter choice.
    ##   p = ncol(bodyfat) - 1 (= Outcome) + 1 ( = Intercept)
    stabsel_parameters(q = 3, PFER = 1, p = ncol(bodyfat) - 1 + 1,
                       sampling.type = "MB")
    ## the same:
    stabsel(mod, q = 3, PFER = 1, sampling.type = "MB", eval = FALSE)

    ## now run stability selection
    (sbody <- stabsel(mod, q = 3, PFER = 1, sampling.type = "MB"))
    opar <- par(mai = par("mai") * c(1, 1, 1, 2.7))
    plot(sbody)
    par(opar)

    plot(sbody, type = "maxsel", ymargin = 6)
}
```

`stabsel_parameters` *Compute Error Bounds for Stability Selection*

Description

Compute

Usage

```
## generic function to compute missing parameter from the other two parameters;
## (internally used within stabsel)
stabsel_parameters(p, ...)

## Default S3 method:
stabsel_parameters(p, cutoff, q, PFER,
                   B = ifelse(sampling.type == "MB", 100, 50),
                   assumption = c("unimodal", "r-concave", "none"),
                   sampling.type = c("SS", "MB"),
                   verbose = FALSE, FWER, ...)
```

Arguments

p	number of possible predictors (including intercept if applicable).
cutoff	cutoff between 0.5 and 1. Preferably a value between 0.6 and 0.9 should be used.
q	number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
PFER	upper bound for the per-family error rate. This specifies the amount of falsely selected base-learners, which is tolerated. See details.
B	number of subsampling replicates. Per default, we use 50 complementary pairs for the error bounds of Shah & Samworth (2013) and 100 for the error bound derived in Meinshausen & Buehlmann (2010). As we use B complementray pairs in the former case this leads to $2B$ subsamples.
assumption	Defines the type of assumptions on the distributions of the selection probabilities and simultaneous selection probabilities. Only applicable for `sampling.type = "SS"`. For `sampling.type = "MB"` we always use code"none".
sampling.type	use sampling scheme of of Shah & Samworth (2013), i.e., with complementarty pairs (`sampling.type = "SS"`), or the original sampling scheme of Meinshausen & Buehlmann (2010).
verbose	logical (default: `TRUE`) that determines wether `warnings` should be issued.
FWER	deprecated. Only for compatibility with older versions, use PFER instead.
...	additional arguments to be passed to next function.

Details

This function implements the error bounds for stability selection by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013).

Two of the three arguments `cutoff`, `q` and `PFER` *must* be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where V is the number of false positives, is bounded by the argument `PFER`.

For more details see also `stabsel`.

Value

An object of class `stabsel_parameters` with a special `print` method. The object has the following elements:

cutoff	cutoff used.
q	average number of selected variables used.
PFER	per-family error rate.
sampling.type	the sampling type used for stability selection.
assumption	the assumptions made on the selection probabilities.

References

N. Meinshausen and P. Buehlmann (2010), Stability selection. *Journal of the Royal Statistical Society, Series B*, **72**, 417–473.

R.D. Shah and R.J. Samworth (2013), Variable selection with error control: another look at stability selection. *Journal of the Royal Statistical Society, Series B*, **75**, 55–80.

See Also

For more details see also `stabsel`.

`subsample` *Draw Random Subsamples*

Description

Set up weight matrix for subsampling with sample proportion $1/2$ to be used with `stabsel`.

Usage

```
subsample(weights, B = 100, strata = NULL)
```

Arguments

`weights`	a numeric vector of weights for the model to be cross-validated.
`B`	number of folds, per default 25 for `bootstrap` and `subsampling` and 10 for `kfold`.
`strata`	a factor of the same length as `weights` for stratification.

Details

The function `subsample` can be used to build an appropriate weight matrix to be used with `stabsel`. See there for more details.

If `strata` is defined sampling is performed in each stratum separately thus preserving the distribution of the `strata` variable in each fold.

See Also

`stabsel`

Examples

```
## just a low-dimensional example
subsample(weights = rep(1, 10), B = 50)
```

Index