Package ‘primr’

November 15, 2019

Title Patient rule-induction method

Version 0.2.0

Description Provides function for performing the patient rule-induction method (PRIM). Allows peeling, pasting and trajectory analysis.

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat (>= 2.1.0),
knitr,
rmarkdown,
fields

VignetteBuilder knitr

RoxygenNote 6.1.1

Imports graphics,
stats

R topics documented:

construct_objfun — *Construct an objective function from a user given one.*

Description

Construct an objective function from a user given one.

Usage

```
construct_objfun(fun)
```

Arguments

`fun`	The function given to `obj.fun` in the main function.

Details

The purpose is to create an objective function that takes two arguments: `y` for the response vector and `inbox` a logical vector indicating if each observation lies in the box for which the objective function is computed.

cv.trajectory — *Cross-validated peeling trajectory*

Description

Performs k-fold cross-validation on peeling for choosing the stopping criterion.

Usage

```
cv.trajectory(y, x, grid = NULL, folds = NULL, nfolds = 10, ...)
```

Arguments

`y`	Numeric vector of response values.
`x`	Numeric or categorical data.frame of input values.
`grid`	Vector of stopping supports for peeling trajectory prediction. If NULL (the default), an initial peeling is carried out on the whole data and its `support` element is used.
`folds`	An integer vector giving the fold index of each observation. If NULL (the default) `nfolds` folds are randomly generated. Directly using `folds` is useful for nonstandard folds such as blocks. Note that `folds` is recycled if necessary.
`nfolds`	Integer giving the number of folds to create if `folds` is NULL.
`...`	Additional arguments to be passed to `peeling`.

Details

The `cv.trajectory` function splits the provided data into `nfolds` several folds. The peeling is carried out on `nfolds -1` folds and the objective function is computed on the remaining fold. This process is repeated excluding each fold successively and the resulting trajectories are averaged at each value in `grid`.

Folds can be given either directly through the argument `folds` or randomly generated using the argument `nfolds`.

Value

A `cv.prim` object that can be used in methods for `prim` objects (e.g. `plot_trajectory`). Contains the elements:

- `support` The support grid provided in the argument `grid` or generated if the latter is `NULL`.
- `yfun` The cross-validated objective function values at each `support` value.
- `se.yfun` Cross-validation standard errors associated with `yfun` values.
- `x,y` The input and response data used.
- `numeric.vars` A logical vector indicating, for each input variable, if it was considered as a numeric variable.
- `alpha, peeling.side, obj.fun` The value of the arguments used for peeling. Useful for prim methods.

References

Friedman, J.H., Fisher, N.I., 1999. Bump hunting in high-dimensional data. Statistics and Computing 9, 123-143. https://doi.org/10.1023/A:1008894516817

See Also

`peeling` for the peeling algorithm used in the function. `plot_trajectory` to analyse the cross-validated trajectory.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) + rnorm(1000)

# 10-fold cross-validation
cv_res <- cv.trajectory(y, x)

# Display the cross-validated trajectory
plot_trajectory(cv_res, type = "b", pch = 16, col = "cornflowerblue",
  support = 0.1, npeel = which.max(cv_res$yfun),
  abline.pars = list(lwd = 2, col = "indianred"),
  xlab = "", xlim = c(0, 0.2), ylim = c(10, 18))
```

`extract.box` *Extract boxes*

Description

Extracts one or several boxes from a `prim` object.

Usage

```
extract.box(object, npeel = NULL, support = NULL, yfun = NULL,
  npaste = NULL)
```

Arguments

`object`	`prim` object.
`npeel`	Integer vector indicating which boxes to choose in `object` through the number of peeling iteration.
`support`	Numeric vector with values between 0 and 1 indicating the support of boxes to choose in `object`.
`yfun`	Numeric vector indicating the value of the objective function for the chosen boxes in `object`.
`npaste`	Integer vector indicating which boxes to choose in `object` through the number of pasting iteration.

Details

Returns one or several boxes from `object`. The boxes can be given through the number of peeling iterations (argument `npeel`), pasting iterations (argument `npaste`), closest support (argument `support`) or closest objective function value (argument `yfun`). Note that several of these arguments can be used at once to return several boxes. If no box matches the arguments or if they are all `NULL`, the last box in `object` is returned.

Value

A list with elements:

`limits`	A list giving the limits of extracted boxes. Each limit is a list with one element per input variable.
`npeel`	A vector giving the number of peeling iterations leading to each extracted box.
`yfun`	A vector giving the objective function value of each extracted box.
`support`	A vector giving the support of each extracted box.

See Also

`peeling` to perform the peeling and create a `prim` object.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) + rnorm(1000)
# Peeling step
peel_res <- peeling(y, x)
# Extracting a single box
single_box <- extract.box(peel_res, support = 0.1)
# Extracting all boxes from a prim object
all_boxes <- extract.box(peel_res, npeel = 0:peel_res$npeel)
```

`jump.prim` *Peeling trajectory jump*

Description

Identifies a jump in the peeling trajectory of `object`.

Usage

```
jump.prim(object, rel.support = TRUE)
```

Arguments

`object`	A `prim` object resulting from a call to `peeling`.
`rel.support`	Logical indicating if the trajectory difference should be relative to the support for finding the jump (default to TRUE).

Details

Computes the (relative) trajectory differences of `object`:

$$\frac{yfun[k] - yfun[k-1]}{support[k-1] - support[k]}$$

and returns its maximum value. The rationale is that the biggest jump in peeling trajectory gives a good cut-off point for the peeling algorithm. If `rel.support` = `FALSE`, the denominator is not used in the differences calculation.

Value

A list with elements:

`trajectory.difference`	Numeric vector of the computed (relative) differences.
`npeel.opt`	Integer giving the npeel value of the highest difference.
`final.box`	The extracted box corresponding to `npeel.opt`. See `extract.box`.

References

Masselot P., Chebana F., Campagna C., Lavigne E., Ouarda T.B.M.J., Gosselin P. On threshold identification for weather-health warning systems. *Submitted*.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) +
  rnorm(1000)
# Peeling with alpha = 0.05 and beta.stop = 0.05
peel_res <- peeling(y, x, beta.stop = 0.05)
# Automatically choose the best box
chosen <- jump.prim(peel_res)
```

`pasting` *Bottom up pasting*

Description

Refines the edges of a chosen box in a `prim` object by pasting.

Usage

```
pasting(object, npeel = NULL, support = NULL, yfun = NULL,
  alpha = object$alpha, obj.fun = object$obj.fun,
  peeling.side = object$peeling.side)
```

Arguments

`object`	`prim` object resulting from a call to `peeling`.
`npeel`	Numeric value indicating which box to choose in `object` through the number of peeling iteration.
`support`	Numeric value between 0 and 1 indicating the support of the box to choose in `object`.
`yfun`	Numeric value indicating the value of the objective function for the chosen box in `object`.
`alpha`	The proportion of observations to add at each pasting iteration. Usually equal to the peeling fraction used in `peeling`.
`obj.fun`	The objective function to maximize by pasting. See `peeling`.
`peeling.side`	Constraints on the pasting side. -1 indicates pasting only on the 'left' of the box (i.e. moving the lower limit only), 1 indicate pasting only on the 'right' and 0 for no constraint.

Details

The function takes a `prim` object and choose one of its boxes as a starting point for pasting. Bottom-up pasting is the reverse of of the top down peeling. It starts from the result of the peeling and refines its edges by iteratively adding `alpha` times N observations at each iteration, where N is the number of observations in the current box.

The best box after the peeling should be chosen by analyzing the peeling trajectory (see `plot_trajectory`). It is given by one of: number of peeling iteration leading to the box (argument `npeel`), the closest support (argument `support`), or the closest objective function value (argument `yfun`).

Although it is possible to use different algorithm parameters (`alpha`, `obj.fun`, `peeling.side`) than the peeling step, it is advised to keep the same values (the default).

Value

A `prim` object which is a list with the following elements:

- `npeel` — The number of peeling iteration performed.
- `support` — A vector of length `npeel + npaste + 1` containing the support of each successivepeeled box.
- `yfun` — A vector of length `npeel + npaste + 1` containing the objective function value of each successive peeled box.
- `limits` — A list of length `npeel + npaste + 1` containing the limits of each successive box. Each limit is a list with one element per input variable.
- `x,y` — The input and response data used in the algorithm.
- `numeric.vars` — A logical vector indicating, for each input variable, if it was considered as a numeric variable.
- `alpha, peeling.side, obj.fun` — The value of the arguments used for peeling. Useful for prim methods.
- `npaste` — Number of pasting iteration performed.

References

Friedman, J.H., Fisher, N.I., 1999. Bump hunting in high-dimensional data. Statistics and Computing 9, 123-143. https://doi.org/10.1023/A:1008894516817

See Also

`extract.box` to extract information about a particular box in a `prim` object. `plot_box` to visualize boxes. `predict.prim` to predict if new data falls into particular boxes.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) + rnorm(1000)
# Peeling step
peel_res <- peeling(y, x)
# Pasting from the box with support 0.01
paste_res <- pasting(peel_res, support = 0.01)
# Visualize the peeled box and pasted one (npaste 0 and 2)
plot_box(paste_res, pch = 16, ypalette = hcl.colors(10), npaste = c(0, 2),
  box.args = list(lwd = 2, border = c("grey", "black"), lty = 1:2))
```

`peeling` *Top-down peeling*

Description

Iteratively peels a dataset for bump hunting.

Usage

```
peeling(y, x, alpha = 0.05, beta.stop = 0.01, obj.fun = mean,
  peeling.side = 0)
```

Arguments

`y`	Numeric vector of response values.
`x`	Numeric or categorical data.frame of input values.
`alpha`	The peeling fraction of the algorithm. A value between 0 and 1 giving the proportion of peeled observations at each step.
`beta.stop`	The stopping support of the algorithm. A value between 0 and 1 giving the proportion of remaining data below which the algorithm stops.
`obj.fun`	The function of `y` to be maximized. Can be a user defined function (see details).
`peeling.side`	A numeric vector for side constraints on the peeling of each input variable. -1 indicates peeling only the 'left' of the box (i.e. increasing the lower limit only), 1 indicate peeling only the 'right' and 0 for no constraint.

Details

The function `peeling` carries out the top-down peeling which is the first step of the PRIM algorithm. At each iteration it peels a proportion `alpha` of data from one side of the domain in order to increase the value of the function `obj.fun` applied to the response `y`. The algorithm iterates the peeling until the support of the box (i.e. the proportion of remaining observations) is below the value `beta.stop`.

Many function can be used in `obj.fun` including user defined functions. User defined function should take two arguments: `y` and `x` representing corresponding variables and `inbox` which is a boolean vector indicating the observations inside the current box. Note that a classical function can also be passed to `obj.fun` such as `mean`, `var` or `median`. In this case the function is created internally to fit the above structure. For more functions more complicated than the basic ones, it is recommended that the user set its own function as stated above.

The function also allows directed peeling, i.e. to contraint the peeling occuring on a single side of some input variables. Thus when `peeling.side = -1`, only the lower part of the variable is peeled (the "left" of the domain) and when `peeling.side = 1`, only the upper part of the variable is peeled. Note that a vector can be passed, thus applying different constraints to the input variables.

Value

A `prim` object which is a list with the following elements:

`npeel`	The number of peeling iteration performed.
`support`	A vector of length `npeel + 1` containing the support of each successivepeeled box.
`yfun`	A vector of length `npeel + 1` containing the objective function value of each successive peeled box.
`limits`	A list of length `npeel + 1` containing the limits of each successive box. Each limit is a list with one element per input variable.
`x,y`	The input and response data used in the algorithm.
`numeric.vars`	A logical vector indicating, for each input variable, if it was considered as a numeric variable.

`alpha, peeling.side, obj.fun`
: The value of the arguments used for peeling. Useful for prim methods.

`npaste`
: Number of pasting iteration performed. Should be 0 here, but useful for `pasting`.

Note that the first box in a `prim` object is the starting box containing the whole dataset. This is why the `limits`, `yfun` and `support` elements have length `npeel` + 1.

References

Friedman, J.H., Fisher, N.I., 1999. Bump hunting in high-dimensional data. Statistics and Computing 9, 123-143. https://doi.org/10.1023/A:1008894516817

See Also

`extract.box` to extract information about a particular box in a `prim` object. `plot_trajectory` and `plot_box` to explore the peeling trajectory. `jump.prim` to automatically choose the best box. `predict.prim` to predict if new data falls into particular boxes. `pasting` to carry out the pasting refining the edges of the chosen box.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) +
  rnorm(1000)
# Peeling with alpha = 0.05 and beta.stop = 0.05
peel_res <- peeling(y, x, beta.stop = 0.05)
# Automatically choose the best box
chosen <- jump.prim(peel_res)
# Plot the resulting box
plot_box(peel_res, pch = 16, ypalette = hcl.colors(10),
  support = chosen$final.box$support, box.args = list(lwd = 2))

# Examples of directed peeling
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 10 * (x[,1] <= .2 & x[,2] <= .2) + 10 * (x[,1] >= .8 & x[,2] >= .8) +
  rnorm(1000)
# Left peeling
peel_left <- peeling(y, x, peeling.side = -1)
chosen <- jump.prim(peel_left)
plot_box(peel_left, pch = 16, ypalette = hcl.colors(10),
  support = chosen$final.box$support, box.args = list(lwd = 2),
  main = "Left peeling")
# Right peeling
peel_right <- peeling(y, x, peeling.side = 1)
chosen <- jump.prim(peel_right)
plot_box(peel_right, pch = 16, ypalette = hcl.colors(10),
  support = chosen$final.box$support, box.args = list(lwd = 2),
  main = "Right peeling")

# User-defined objective function to minimize the mean
set.seed(3333)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- - 10 * (x[,1] <= .2 & x[,2] <= .2) + 10 * (x[,1] >= .8 & x[,2] >= .8) +
```

```
  rnorm(1000)
peel_res <- peeling(y, x, obj.fun = function(x) -mean(x))
chosen <- jump.prim(peel_res)
plot_box(peel_res, pch = 16, ypalette = hcl.colors(10),
  support = chosen$final.box$support, box.args = list(lwd = 2))

# User-defined function maximizing the slope of a linear regression
set.seed(5555)
x <- runif(500)
ym <- 0.5 * x + 5 * (x - 0.7) * (x >= 0.7)
y <- ym + rnorm(500, sd = 0.1)
peel_res <- peeling(y, x, beta.stop = 0.1,
  obj.fun = function(y, x, inbox){
    dat <- data.frame(y, x)
    coef(lm(y ~ x, data = dat[inbox,]))[2]
})
par(mfrow = c(1,2))
plot_trajectory(peel_res, type = "b", pch = 16, col = "cornflowerblue",
  support = 0.3, abline.pars = list(lwd = 2, col = "indianred"))
plot_box(peel_res, pch = 16, ypalette = hcl.colors(10),
  support = 0.3, box.args = list(lwd = 2))
lines(sort(x), ym[order(x)], col = "red", lwd = 2)
```

`plot_box` *Plot a box*

Description

Plots a bidimensional projection of the data and chosen boxes.

Usage

```
plot_box(object, select = 1:2, npeel = NULL, support = NULL,
  yfun = NULL, npaste = NULL, ypalette = NULL, box.args = list(),
  ...)
```

Arguments

`object`	A 'prim' object.
`select`	A vector of length 1 or 2 indicating the input variables(s) to plot. Default to the two first ones. If a single variable is selected, plot it against `y`.
`npeel`	Integer vector indicating the number of peeling iteration of boxes to plot.
`support`	Numeric vector with values between 0 and 1 indicating the support of boxes to plot.
`yfun`	Numeric vector indicating the value of the objective function of the boxes to plot.
`npaste`	Integer vector indicating the number of pasting iteration of boxes to plot.
`ypalette`	A palette of colors for representing the `y` value associated to each point.
`box.args`	A list of arguments to be passed to `rect` for drawing boxes. All arguments can be given as vectors to specify different values for each drawn box.
`...`	Additional graphical parameters for `plot`.

Details

Several boxes can be displayed at once, selected by one of the arguments `npeel`, `support`, `yfun` or `npaste` (when relevant). Note that the arguments in `box.args` allow giving different parameters to each displayed box.

See Also

`peeling` for peeling trajectories.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) +
  rnorm(1000)
# Peeling with alpha = 0.05 and beta.stop = 0.05
peel_res <- peeling(y, x, beta.stop = 0.05)
# Automatically choose the best box
chosen <- jump.prim(peel_res)
# Plot the resulting box
plot_box(peel_res, pch = 16, ypalette = hcl.colors(10),
  support = chosen$final.box$support, box.args = list(lwd = 2))
```

`plot_trajectory` *Plot a peeling trajectory*

Description

Displays the peeling trajectory of a `prim` object with chosen cut-points.

Usage

```
plot_trajectory(object, xtype = c("support", "nobs"), ytype = c("yfun",
  "diff", "rel.diff"), npeel = NULL, support = NULL, yfun = NULL,
  npaste = NULL, abline.pars = list(), se = TRUE, se.pars = list(),
  ...)
```

Arguments

`object`	A `prim` object.
`xtype`	A character indicating the how to display the x axis of the plot. 'support' (the default) for the support of the boxes or 'nobs' for the number of observations inside the box.
`ytype`	A character indicating which trajectory to plot. 'yfun' (the default) for the objective function value, 'diff' for the difference between successive boxes and 'rel.diff' for the relative difference. See `jump.prim` for details.
`npeel`	Integer vector. If not `NULL`, draw a vertical line at the corresponding peeling iterations of the trajectory.

support	Numeric vector. If not NULL, draw a vertical line at the corresponding supports of the trajectory.
yfun	Numeric vector. If not NULL, draw a vertical line at the boxes with the closest yfun value.
npaste	Integer vector. If not NULL, draw a vertical line at the corresponding pasting iterations of the trajectory.
abline.pars	List of parameters to be passed to abline for the vertical lines.
se	Logical indicating if standard error bars should be drawn. Works only for cv.prim objects and ytype = "yfun".
se.pars	Parameters to be passed to arrows for drawing strandard error bars.
...	Additional graphical parameters for plot.

References

Friedman, J.H., Fisher, N.I., 1999. Bump hunting in high-dimensional data. Statistics and Computing 9, 123-143. https://doi.org/10.1023/A:1008894516817

See Also

peeling for peeling trajectories.

Examples

```
# A simple bump
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) +
  rnorm(1000)
# Peeling with alpha = 0.05 and beta.stop = 0.05
peel_res <- peeling(y, x, beta.stop = 0.05)

# Display the peeling trajectory
plot_trajectory(peel_res, type = "b", pch = 16, col = "cornflowerblue",
  support = 0.11, abline.pars = list(lwd = 2, col = "indianred"),
  xlab = "")
```

---

predict.prim *Predict method for a* prim *object*

Description

For each observation in newx, check if it is in selected boxes of object. Also gives the objective function values of each box based on the newy values.

Usage

```
## S3 method for class 'prim'
predict(object, newx, newy, npeel = NULL,
  support = NULL, yfun = NULL, npaste = NULL, ...)
```

Arguments

`object`	A `prim` object.
`newx`	A data.frame of new input values. If missing, uses `object$x`.
`newy`	A vector of new responses values corresponding to the inputs in `newx`. If missing, uses `object$y`.
`npeel`	Integer vector indicating which boxes in `object` to use for prediction, through the number of peeling iteration.
`support`	Numeric vector with values between 0 and 1 indicating the support of boxes in `object` to use for prediction.
`yfun`	Numeric vector indicating the value of the objective function for the boxes in `object` to use for prediction.
`npaste`	Integer vector indicating which boxes in `object` to use for prediction, through the number of pasting iteration.
`...`	further arguments passed to or from other methods.

Value

A list with elements:

`inbox`	A logical matrix of dimension n x nboxes where n is the number of observations in `newx`. For each observation and each returns `TRUE` if the observation is inside the box.
`support`	A numeric vector giving, for each box, the proportion of observations in `newx` lying inside the box.
`yfun`	A numeric vector giving, for each box, the objective function value computed on `newy`.

See Also

`peeling` and `pasting` for creating `prim` objects.

Examples

```
set.seed(12345)
x <- matrix(runif(2000), ncol = 2, dimnames = list(NULL, c("x1", "x2")))
y <- 2 * x[,1] + 5 * x[,2] + 10 * (x[,1] >= .8 & x[,2] >= .5) + rnorm(1000)
# Peeling step
peel_res <- peeling(y, x)
# Prediction
predict(peel_res, newx = data.frame(x1 = 1:10/11, x2 = 1:10/11),
  newy = 1:10, support = c(1, 0.25))
```

Index