Package 'mmpf'

October 24, 2018

Title Monte-Carlo Methods for Prediction Functions

Description Marginalizes prediction functions using Monte-Carlo integration and computes permutation importance.

Version 0.0.5

Date 2018-10-23

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

Imports checkmate, data.table

Suggests testthat, rmarkdown, knitr, randomForest, ggplot2, reshape2

BugReports https://github.com/zmjones/mmpf

VignetteBuilder knitr

NeedsCompilation no

Author Zachary Jones [aut, cre]

Maintainer Zachary Jones <zmj@zmjones.com>

Repository CRAN

Date/Publication 2018-10-24 04:20:26 UTC

R topics documented:

cartesianExpand *expands two data.frames using the Cartesian product*

Description

takes the cartesian product of two data.frames

Usage

```
cartesianExpand(x, y)
```

Arguments

x	a data.frame
y	a data.frame

Value

a data.frame

Examples

```
x = data.frame("a" = 1:5, "b" = 6:10)
y = data.frame("z" = letters[1:5], "y" = letters[6:10])
cartesianExpand(x, y)
```

makeDesign *make a uniform, random, or user-specified grid over some columns of a data.frame, and combine it with a grid of points to integrate over.*

Description

makes a uniform, random, or user-specified grid over some columns of a data.frame and takes their Cartesian product with the other columns

Usage

```
makeDesign(data, vars, n, uniform = TRUE, points, int.points)
```

Arguments

data	a data.frame which must contain vars as well as at least one other column
vars	character vector the columns in data to create the grid for
n	two dimensional integer vector giving the resolution of the grid. the first element gives the grid on vars and the second on the other columns, which are sampled without replacement.
uniform	logical, indicates whether a uniform grid is to be constructed.
points	a named list which gives specific points for vars.
int.points	a integer vector giving indices of the points in data to marginalize over.

Value

a data.frame with at most n dimensions.

Examples

```
data = data.frame(w = seq(0, 1, length.out = 5),
  x = factor(letters[1:5]),
  y = ordered(1:5),
  z = 1:5,
  r = letters[1:5],
  stringsAsFactors = FALSE)
makeDesign(data, "z", c(10, 5), TRUE)
```

---

makePermutedDesign *creates a* data.frame *with some columns permuted*

Description

takes an input data.frame, permutes some variables, and stacks the resulting data.frames.

Usage

```
makePermutedDesign(data, vars, nperm)
```

Arguments

data	a data.frame a subset of which must be vars.
vars	a character vector indicating columns in data to permute.
nperm	an integer specifying the number of times to permute the columns indicated by vars.

Value

a data.frame with number of rows equal to nrow(data) * nperm

Examples

```
data = data.frame(x = 1:3, y = letters[1:3])
makePermutedDesign(data, "x", 3)
```

`marginalPrediction` *marginalizes prediction functions*

Description

monte-carlo integration of prediction functions

Usage

```
marginalPrediction(data, vars, n, model, uniform = TRUE, points,
  int.points, aggregate.fun = function(x) sum(x)/length(x),
  predict.fun = function(object, newdata) predict(object, newdata =
  newdata), weight.fun = NULL)
```

Arguments

`data`	a `data.frame` which contains the columns specified by `vars` and at least one additional column. should correspond to the set of columns used to train the `model`.
`vars`	a character vector corresponding to a strict subset of the columns in `data`.
`n`	an integer vector of length two giving the resolution of the uniform or random grid on `vars` for the first element, and the number of the rows of the `data` to be sampled without replacement for the second element.
`model`	an object which can be passed to `predict.fun` to compute predictions. presumably this object represents a model fit.
`uniform`	logical indicating whether to create the grid on `vars` uniformly or to sample without replacement from the empirical distribution of those `vars`.
`points`	a named list which gives specific points for `vars`. specifying this argument overrides `uniform`.
`int.points`	a integer vector giving indices of the points in `data` to marginalize over.
`aggregate.fun`	what function to aggregate the predictions with. this function takes a single argument `x` and returns a vector. the default is `sum(x) / length(x)`. If `weight.fun` is used, this function must also take a numeric parameter `w`.
`predict.fun`	what function to generate predictions using `model`. default is the predict method for `model`. this function must have two arguments, `object` and `newdata`.
`weight.fun`	a function to construct weights for `aggregate.fun`. this allows Monte-Carlo integration on a grid without assuming a uniform distribution for said grid. the function should take two arguments, `design` and `data`, both of which are `data.frames` of the same column (but different row) dimension, and should return a numeric vector of the same length as the number of rows in `design`. If this argument is used `aggregate.fun` must also have an argument `w` which is the result of `weight.fun`.

Value

a data.table with columns for predictions and vars.

Examples

```
X = replicate(3, rnorm(100))
y = X %*% runif(3)
data = data.frame(X, y)
fit = lm(y ~ ., data)

marginalPrediction(data.frame(X), "X2", c(10, 25), fit,
  aggregate.fun = function(x) c("mean" = mean(x), "variance" = var(x)))
```

---

permutationImportance *computes permutation importance*

Description

computes the change in prediction error from permuting variables.

Usage

```
permutationImportance(data, vars, y, model, nperm = 100L,
  predict.fun = function(object, newdata) predict(object, newdata =
  newdata), loss.fun = function(x, y) defaultLoss(x, y),
  contrast.fun = function(x, y) x - y)
```

Arguments

data	a data.frame including both y and vars.
vars	a character vector specifying columns of data to permute.
y	a character vector giving the name of the target/outcome variable.
model	an object with a predict method which returns a vector or matrix. presumably this object represents a model fit.
nperm	positive integer giving the number of times to permute the indicated variables (default is 100).
predict.fun	what function to generate predictions using model. default is the predict method for model. the function must take two arguments, object and newdata and should return a vector or matrix.
loss.fun	what loss function to use to measure prediction errors. default is mean squared-error for ordered predictions and mean misclassification error for unordered prediction errors. this function must take two arguments, "x" and "y", which operate on the output of predict.fun and data[, y].
contrast.fun	what function to use to contrast the permuted and unpermuted predictions. default is the difference. this function takes two arguments "x" and "y", which are the output of the loss.fun.

Value

a numeric vector or matrix, depending on `contrast.fun` and `loss.fun`, giving the change in prediction error from `nperm` permutations of `vars`.

Examples

```
X = replicate(3, rnorm(100))
y = X %*% runif(3)
data = data.frame(X, y)
fit = lm(y ~ -1 + X1 + X2 + X3, data)

permutationImportance(data, "X1", "y", fit)
```

`uniformGrid` *method to create a uniform grid on a variable*

Description

generates an evenly spaced grid given an input vector, matrix, or `data.frame` which has size `length.out`.

Usage

```
uniformGrid(x, length.out)
```

Arguments

`x`	a vector, matrix, or `data.frame` to create a grid on.
`length.out`	an integer giving the length of the grid.

Value

an object of the same type as `x`, with `length.out` or fewer unique values.

Note

for unordered factors and characters, if `length.out < length(unique(x))` `length.out` is set to `length(unique(x))`. if `x` is a `data.frame` and this is true of some columns but not others, there will be a warning.

Examples

```
data = data.frame(
  w = seq(0, 1, length.out = 5),
  x = factor(letters[1:5]),
  y = ordered(1:5),
  z = 1:5
```

```
)

lapply(data, uniformGrid, length.out = 5)
```

Index