Package 'FOCI'

May 16, 2020

Type Package

Title Feature Ordering by Conditional Independence

Version 0.1.2

Maintainer Mona Azadkia <monaazadkia@gmail.com>

Description Feature Ordering by Conditional Independence (FOCI) is a variable selection algorithm based on the measure of conditional dependence.
For more information, see the paper: Azadkia and Chatterjee (2019),``A simple measure of conditional dependence'' <arXiv:1910.12327>.

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.1.0

Suggests knitr, rmarkdown, testthat

Depends R (>= 3.6.0), data.table

Imports RANN, proxy, parallel, gmp

NeedsCompilation no

Author Mona Azadkia [aut, cre],
Sourav Chatterjee [aut, ctb],
Norman Matloff [aut, ctb]

Repository CRAN

Date/Publication 2020-05-16 05:40:07 UTC

R topics documented:

codec *Estimate the conditional dependence coefficient (CODEC)*

Description

The conditional dependence coefficient (CODEC) is a measure of the amount of conditional dependence between a random variable Y and a random vector Z given a random vector X, based on an i.i.d. sample of (Y, Z, X). The coefficient is asymptotically guaranteed to be between 0 and 1.

Usage

```
codec(Y, Z, X = NULL, na.rm = TRUE)
```

Arguments

Y	Vector (length n)
Z	Matrix (n by q)
X	Matrix (n by p), default is NULL
na.rm	Remove NAs if TRUE

Details

The value returned by codec can be positive or negative. Asymptotically, it is guaranteed to be between 0 and 1. A small value indicates low conditional dependence between Y and Z given X, and a high value indicates strong conditional dependence. The codec function is used by the foci function for variable selection.

Value

The conditional dependence coefficient (CODEC) of Y and Z given X. If X == NULL, this is just a measure of the dependence between Y and Z.

Author(s)

Mona Azadkia, Sourav Chatterjee, Norman Matloff

References

Azadkia, M. and Chatterjee, S. (2019). A simple measure of conditional dependence. https://arxiv.org/pdf/1910.12327.pdf.

See Also

foci, xicor

Examples

```
n = 1000
x <- matrix(runif(n * 2), nrow = n)
y <- (x[, 1] + x[, 2]) %% 1
# given x[, 1], y is a function of x[, 2]
codec(y, x[, 2], x[, 1])
# y is a function of x
codec(y, x)
z <- rnorm(n)
# y is a function of x given z
codec(y, x, z)
# y is independent of z given x
codec(y, z, x)
```

`foci` *Variable selection by the FOCI algorithm*

Description

FOCI is a variable selection algorithm based on the measure of conditional dependence `codec`.

Usage

```
foci(
  Y,
  X,
  num_features = NULL,
  stop = TRUE,
  na.rm = TRUE,
  standardize = "scale",
  numCores = parallel::detectCores(),
  parPlat = "none",
  printIntermed = TRUE
)
```

Arguments

`Y`	Vector of responses (length n)
`X`	Matrix of predictors (n by p)
`num_features`	Number of variables to be selected, cannot be larger than p. The default value is NULL and in that case it will be set equal to p. If stop == TRUE (see below), then num_features is irrelevant.
`stop`	Stops at the first instance of negative codec, if TRUE.
`na.rm`	Removes NAs if TRUE.

standardize	Standardize covariates if set equal to "scale" or "bounded". Otherwise will use the raw inputs. The default value is "scale" and normalizes each column of X to have mean zero and variance 1. If set equal to "bounded" map the values of each column of X to [0, 1].
numCores	Number of cores that are going to be used for parallelizing the variable selecction process.
parPlat	Specifies the parallel platform to chunk data by rows. It can take three values: 1- The default value is set to 'none', in which case no row chunking is done; 2- the parallel cluster to be used for row chunking; 3- "locThreads", specifying that row chunking will be done via threads on the host machine.
printIntermed	The default value is TRUE, in which case print intermediate results from the cluster nodes before final processing.

Details

FOCI is a forward stepwise algorithm that uses the conditional dependence coefficient (codec) at each step, instead of the multiple correlation coefficient as in ordinary forward stepwise. If stop == TRUE, the process is stopped at the first instance of nonpositive codec, thereby selecting a subset of variables. Otherwise, a set of covariates of size num_features, ordered according to predictive power (as measured by codec) is produced.

Parallel computation:

The computation can be lengthy, so the package offers two kinds of parallel computation.

The first, controlled by the argument numCores, specifies the number of cores to be used on the host machine. If at a given step there are k candidate variables under consideration for inclusion, these k tasks are assigned to the various cores.

The second approach, controlled by the argument parPlat ("parallel platform"), involves the user first setting up a cluster via the **parallel** package. The data are divided into chunks by rows, with each cluster node applying FOCI to its data chunk. The union of the results is then formed, and fed through FOCI one more time to adjust the discrepancies. The idea is that that last step will not be too lengthy, as the number of candidate variables has already been reduced. A cluster size of r may actually produce a speedup factor of more than r (Matloff 2016).

Potentially the best speedup is achieved by using the two approaches together.

The first approach cannot be used on Windows platforms, as parallel::mcapply has no effect. Windows users should thus use the second approach only.

In addition to speed, the second approach is useful for diagnostics, as the results from the different chunks gives the user an idea of the degree of sampling variability in the FOCI results.

In the second approach, a random permutation is applied to the rows of the dataset, as many datasets are sorted by one or more columns.

Note that if a certain value of a feature is rare in the full dataset, it may be absent entirely in some chunk.

Value

An object of class "foci", with attributes selectedVar, showing the selected variables in decreasing order of (conditional) predictive power, and stepT, listing the 'codec' values. Typically the latter will begin to level off at some point, with additional marginal improvements being small.

Author(s)

Mona Azadkia, Sourav Chatterjee, and Norman Matloff

References

Azadkia, M. and Chatterjee, S. (2019). A simple measure of conditional dependence. https://arxiv.org/pdf/1910.12327.pdf.

Matloff, N. (2016). Software Alchemy: Turning Complex Statistical Computations into Embarrassingly-Parallel Ones. *J. of Stat. Software.*

See Also

codec, xicor

Examples

```
# Example 1
n = 1000
p = 100
x <- matrix(rnorm(n * p), nrow = n)
colnames(x) = paste0(rep("x", p), seq(1, p))
y <- x[, 1] * x[, 10] + x[, 20]^2
# with num_features equal to 3 and stop equal to FALSE, foci will give a list of
# three selected features
result1 = foci(y, x, num_features = 3, stop = FALSE, numCores = 1)
result1
# Example 2
# same example, but stop according to the stopping rule
result2 = foci(y, x, numCores = 1)
result2
## Not run:
# Windows use of multicore
library(parallel)
cls <- makeCluster(parallel::detectCores())
foci(y, x, parPlat = cls)
# run on physical cluster
cls <- makePSOCKcluster('machineA','machineB')
foci(y, x, parPlat = cls)

## End(Not run)
```

Index