Package 'FSInteract'

July 21, 2025

Type Package

Title Fast Searches for Interactions

Version 0.1.2

Date 2017-04-03

Author Hyun Jik Kim, Rajen D. Shah

Maintainer Rajen D. Shah <r.shah@statslab.cam.ac.uk>

Description Performs fast detection of interactions in large-scale data using the method of random intersection trees introduced in Shah, R. D. and Meinshausen, N. (2014) <http://www.jmlr.org/papers/v15/shah14a.html>. The algorithm finds potentially high-order interactions in high-dimensional binary two-class classification data, without requiring lower order interactions to be informative. The search is particularly fast when the matrices of predictors are sparse. It can also be used to perform market basket analysis when supplied with a single binary data matrix. Here it will find collections of columns which for many rows contain all 1's.

License GPL-2

Imports Rcpp, Matrix, methods

LinkingTo Rcpp

URL http://www.jmlr.org/papers/v15/shah14a.html

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-04-03 21:58:38 UTC

Contents

RIT *Random Intersection Trees*

Description

Function to perform random intersection trees. When two binary data matrices `z` (class 1) and `z0` (class 0) are supplied, it searches for interactions. More precisely, since the data matrices are binary, each row of each matrix can be represented by the set of column indices with non-zero entries. The function searches for sets (interactions) that are more prevalent in class 1 than class 0, and then sets that are more prevalent in class 0 than class 1. When given a single binary matrix `z` with the argument `z0` omitted, the function simply finds sets with high prevalence. Prevalences of interactions returned are estimated using min-wise hashing.

Usage

```
RIT(z, z0, branch = 5, depth = 10L, n_trees = 100L,
    theta0 = 0.5, theta1 = theta0, min_inter_sz = 2L,
    L = 100L, n_cores = 1L, output_list = FALSE)
```

Arguments

`z`	data matrix where each row corresponds to an observation and columns correspond to variables. Can be in sparse matrix format (inherit from class "sparseMatrix" in the **Matrix** package).
`z0`	optional second data matrix with the same number of columns as `z`.
`branch`	average number of branches to use when creating each tree.
`depth`	maximum depth of trees.
`n_trees`	number of trees to be constructed.
`theta0`	when searching for sets of variables that are more prevalent in class 1 than class 0, the maximum threshold for prevalence in class 0.
`theta1`	as above but with class 1 and class 0 interchanged.
`min_inter_sz`	minimum size of the interactions to be returned
`L`	number of rows of the min-wise hash matrix used to estimate prevalences. A larger value will result in more accurate estimates, but computation time will increase linearly with `L`.
`n_cores`	number of cores for parallel processing. Only used when openMP is installed.
`output_list`	if `FALSE` returns each interaction set as a string with variable indices separated by spaces. If `TRUE` returns each interaction set as an integer vector.

Details

There are two tasks which can be performed with this function depending on whether or not `z0` is supplied (note `z` must always be supplied).

1. If `z0` is omitted, the function finds prevalent sets in `z` and `theta0` and `theta1` are ignored.

2. If `z0` is supplied, it searches for sets that are prevalent in `z` but have prevalence at most `theta0` in `z0`. Next sets that are prevalent in `z0` but have prevalence in `z` at most `theta1` are found.

Value

If `output_list` is `FALSE` (the default), the output is either a data frame (if `z0` is omitted) or list of two data frames (if `z0` is supplied). The data frames have first column a character vector of interaction sets with the variables in the sets separated by spaces, and second column the estimated prevalences. When `z0` is supplied, the interactions in the first component of the list named `Class1` are those which are prevalent in `z` and their prevalences in `z` are reported. The second component named named `Class0` contains those interactions prevalent in `z0` and their prevalences in `z0`.

When `output_list` is `TRUE`, each interaction is reported as an integer vector and so the collection of interactions is a list of such vectors.

Author(s)

Hyun Jik Kim, Rajen D. Shah

References

Shah, R. D. and Meinshausen, N. (2014) Random Intersection Trees. *Journal of Machine Learning Research*, **15**, 629–654.

Examples

```
## Generate two binary matrices
z <- matrix(rbinom(250*500, 1, 0.3), 250, 500)
z0 <- matrix(rbinom(250*500, 1, 0.3), 250, 500)

## Make the first and second cols of z identical
## so the set 1, 2 has prevalence roughly 0.3 compared
## to roughly 0.09 for any other pair of columns
z[, 1] <- z[, 2]

## Similarly for z0
z0[, 3] <- z0[, 4]

## Market basket analysis
out1 <- RIT(z)
out1[1:5, ]

## Finding interactions
out2 <- RIT(z, z0)
out2$Class1[1:5, ]
out2$Class0[1:5, ]

## Can also perform the above using sparse matrices
if (require(Matrix)) {
  S <- Matrix(z, sparse=TRUE)
  S0 <- Matrix(z0, sparse=TRUE)
  out3 <- RIT(S, S0)
}
```

Index