Package ‘threeboost’

February 20, 2015

Type Package
Title Thresholded variable selection and prediction based on estimating equations
Version 1.1
Date 2014-08-09
Author Julian Wolfson and Christopher Miller
Maintainer Julian Wolfson <julianw@umn.edu>
Description This package implements a thresholded version of the EEBoost algorithm described in [Wolfson (2011, JASA)]. EEBoost is a general-purpose method for variable selection which can be applied whenever inference would be based on an estimating equation. The package currently implements variable selection based on the Generalized Estimating Equations, but can also accommodate user-provided estimating functions. Thresholded EEBoost is a generalization which allows multiple variables to enter the model at each boosting step.
License GPL-3
Imports Matrix
Suggests mvtnorm
NeedsCompilation no
Repository CRAN
Date/Publication 2014-08-11 00:18:02

R topics documented:

threeboost-package .................................................. 2
coe_traceplot ...................................................... 2
ee.GEE ............................................................... 2
eeboost ............................................................... 3
ggeeboost ............................................................ 4
QIC .................................................................. 5
threeboost ............................................................. 6

Index 9
threeboost-package

Thresholded boosting based on estimating equations

Description

The threeboost package implements a the EEBoost algorithm described in Wolfson (2011, JASA). EEBoost is a general-purpose method for variable selection which can be applied whenever inference would be based on an estimating equation. Thresholded EEBoost (function threeboost) is a generalization of EEBoost which allows multiple variables to enter the model at each boosting step. EEBoost (function eeboost) is a special case of thresholded boosting with the threshold set to 1.

The package currently provides a "pre-packaged" function, geeboost, which carries out variable selection for correlated outcome data based on the Generalized Estimating Equations. However, the threeboost (and eeboost) functions can also accommodate user-provided estimating functions.

coeff_traceplot

Draw a coefficient traceplot

Description

This function draws a 'traceplot' of coefficient values vs. number of iterations.

Usage

coeff_traceplot(coef.mat, varnames = NULL)

Arguments

coef.mat The matrix of coefficients (one coefficient vector per row).
varnames (Optional) list of variable name labels to make the traceplot more readable.

ee.GEE

GEE estimating functions

Description

Internal functions for computing the GEE. Should generally not be called by user.

Usage

ee.GEE(Y, X, b, mu.Y, g.Y, v.Y, aux, id = 1:length(Y),
uid = sort(unique(id)), rows.indivs = lapply(uid, function(j) {
which(id == j )}), corstr = "ind")

ee.GEE.aux(Y, X, b, mu.Y, g.Y, v.Y, id = 1:length(Y),
uid = sort(unique(id)), rows.indivs = lapply(uid, function(j) {
which(id == j )}))
Arguments

- `Y` Vector of (correlated) outcomes
- `X` Matrix of predictors
- `b` Vector of coefficients
- `mu.Y` Mean function
- `g.Y` Link function (inverse of mean function)
- `v.Y` Variance function
- `aux` Auxiliary function for computing (co)variance parameters
- `id` Vector of cluster IDs
- `uid` Vector of unique subject IDs
- `rows.indivs` List of rows of `X` corresponding to each subject ID
- `corstr` Working correlation structure

Functions

- `::`

---

**Description**

Alias for ThrEEBoost (which defaults to a threshold value of 1).

**Usage**

```r
eebost(...)```

**Arguments**

... Arguments to `threeboost`.

**See Also**

`threeboost`
geeboost  

**Description**

Thresholded boosting for correlated data via GEE

**Usage**

```r
geeboost(y, X, id = 1:length(y), family = "gaussian", corstr = "ind", traceplot = FALSE, ...)
```

**Arguments**

- `y` Vector of (presumably correlated) outcomes
- `X` Matrix of predictors
- `id` Index indicating clusters of correlated observations
- `family` Outcome distribution to be used. "gaussian" (the default), "binomial", and "poisson" are currently implemented.
- `corstr` Working correlation structure to use. "ind" (for independence, the default) and "exch" (for exchangeable) are currently implemented.
- `traceplot` Option of whether or not to produce a traceplot of the coefficient values. See `coef_traceplot` for details.
- `...` Additional arguments to be passed to the `threeboost` function. See the `threeboost` help page for details.

**Details**

This function implements thresholded EEBest for the Generalized Estimating Equations. The arguments are consistent with those used by geepack.

**Value**

A list with three entries:

- `coefmat` A matrix with `maxit` rows and `ncol(X)` columns, with each row containing the parameter vector from an iteration of EEBest.
- `QICs` A vector of QICs computed from the coefficients.
- `final.model` The coefficients corresponding to the model (set of coefficients) yielding the smallest QIC.

**See Also**

`threeboost`

Examples

# Generate some test data
library(mvtnorm)
library(matrix)
n <- 30
n.var <- 50
clust.size <- 4

B <- c(rep(2.5), rep(0.2,5), rep(0.05,10), rep(0,n.var-20))
mn.X <- rep(0,n.var)
sd.X <- 0.5
rho.X <- 0.3
cov.sig.X <- sd.X^2*((1-rho.X)*diag(rep(1,10)) + rho.X*matrix(data=1,nrow=10,ncol=10))
sig.X <- as.matrix( Matrix::bdiag(lapply(1:(n.var/10),function(x) { cov.sig.X }) ) )
sd.Y <- 0.5
rho.Y <- 0.3
indiv.Sig <- sd.Y^2*( (1-rho.Y)*diag(rep(1,4)) + rho.Y*matrix(data=1,nrow=4,ncol=4) )
sig.list <- list(length=n)
for(i in 1:n) { sig.list[[i]] <- indiv.Sig }
Sig <- Matrix::bdiag(sig.list)
indiv.index <- rep(1:n,each=clust.size)
sig.Y <- as.matrix(Sig)

if(require(mvtnorm)) {
  X <- mvtnorm::rmvnorm(n*clust.size,mean=mn.X,sigma=sig.X)
mn.Y <- X %*% B
  Y <- mvtnorm::rmvnorm(1,mean=mn.Y,sigma=sig.Y) ## Correlated continuous outcomes
  expit <- function(x) { exp(x) / (1 + exp(x)) }
## Correlated binary outcomes
  Y.bin <- rbinom(n*clust.size,1,p=expit(mvtnorm::rmvnorm(1,mean=mn.Y,sigma=sig.Y)))
  Y.pois <- rpois(length(Y),lambda=exp(mn.Y)) ## Correlated Poisson outcomes
} else { stop('Need mvtnorm package to generate correlated data.')}

## Run EEBoost (w/ indep working correlation)
results.lin <- geeboost(Y,X,id=indiv.index,maxit=1000)
## Not run:
  results.bin <- geeboost(Y.bin,X,id=indiv.index,family="binomial",maxit=1000)
  results.pois <- geeboost(Y.pois,X,id=indiv.index,family="poisson",maxit=1000,traceplot=TRUE)
## End(Not run)

print(results.lin$final.model)

QIC

Pan’s QIC

Description

Calculates a simple version of Pan’s QIC for a GEE model defined by a vector of regression coefficients.
Usage
QIC(Y, X, b, family = "gaussian")

Arguments
Y        A vector of outcomes.
X        A matrix of predictors.
b        A vector of regression coefficients (e.g., a row from the coefficient matrix produced by geeboost)
family   Version of QIC to implement, for either "gaussian", "binomial" or "poisson" outcomes. Should match the family argument used in the original boosting algorithm.

threeboost
Thresholded EEBoost

Description
Run the thresholded EEBoost procedure.

Usage
threeboost(Y, X, EE.fn, b.init = rep(0, ncol(X)), eps = 0.01, maxit = 1000, itertrack = FALSE, reportinterval = 1, stop.rule = "on.repeat", thresh = 1)

Arguments
Y        Vector of outcomes.
X        Matrix of predictors. Will be automatically scaled using the scale function.
EE.fn    Estimating function taking arguments Y, X, and parameter vector b.
b.init   Initial parameter values. For variable selection, typically start with a vector of zeroes (the default).
eps      Step length. Default is 0.01, value should be relatively small.
maxit    Maximum number of iterations. Default is 1000.
itertrack Indicates whether or not diagnostic information should be printed out at each iteration. Default is FALSE.
reportinterval If itertrack is TRUE, how many iterations the algorithm should wait between each diagnostic report.
stop.rule Rule for stopping the iterations before maxit is reached. Possible values are "on.repeat" and "pct.change". See 'Details' for more information.
thresh   Threshold parameter for ThrEEBoost.
Details
threeboost Implements a thresholded version of the EEBoost algorithm described in Wolfson (2011, JASA). EEBoost is a general-purpose method for variable selection which can be applied whenever inference would be based on an estimating equation. The package currently implements variable selection based on the Generalized Estimating Equations, but can also accommodate user-provided estimating functions. Thresholded EEBoost is a generalization which allows multiple variables to enter the model at each boosting step. Thresholded EEBoost with thresholding parameter = 1 is equivalent to EEBoost.

Typically, the boosting procedure is run for maxit iterations, producing maxit models defined by a set of regression coefficients. An additional step (e.g. model scoring, cross-validated estimate of prediction error) is needed to select a final model. However, an alternative is to stop the iterations before maxit is reached. The user can request this feature by setting stop.rule to one of the following options:

- "on.repeat": Sometimes, ThrEEBoost will alternate between stepping on the same two directions, usually indicating numerical problems. Setting stop.rule="on.oscillate" will terminate the algorithm if this happens.
- "pct.change": Stop if, for consecutive iterations, the sum of the magnitudes of the elements of the estimating equation changes by < 1%.

Value
A matrix with maxit rows and ncol(X) columns, with each row containing the parameter vector from an iteration of ThrEEBoost.

See Also
ggeeboost for an example of how to call (Thr)EEBoost with a custom estimating function.


Examples
library(Matrix)

# Generate some test data - uses 'mvtnorm' package
n <- 30
n.var <- 50
clust.size <- 4
B <- c(rep(2,5),rep(0.2,5),rep(0.05,10),rep(0,n.var-20))
mn.X <- rep(0,n.var)
sd.X <- 0.5
rho.X <- 0.3
cov.sig.X <- sd.X^2*((1-rho.X)*diag(rep(1,10)) + rho.X*matrix(data=1,nrow=10,ncol=10))
sig.X <- as.matrix( Matrix::bdiag(lapply(1:(n.var/10),function(x) { cov.sig.X } ) ) )
sd.Y <- 0.5
rho.Y <- 0.3
indiv.Sig <- sd.Y^2* ( (1-rho.Y)*diag(rep(1,4)) + rho.Y*matrix(data=1,nrow=4,ncol=4) )
sig.list <- list(length=n)
for(i in 1:n) { sig.list[[i]] <- indiv.Sig }
Sig <- Matrix::bdiag(sig.list)
indiv.index <- rep(1:n,each=clust.size)
sig.Y <- as.matrix(Sig)
if(require(mvtnorm)) {
  X <- mvtnorm::rmvnorm(n*clust.size,mean=mn.X,sigma=sig.X)
  mn.Y <- X %*% B
  ## Correlated continuous outcome
  Y <- mvtnorm::rmvnorm(1,mean=mn.Y,sigma=sig.Y)
} else { stop('Need mvtnorm package to generate correlated example data.') }

## Define the Gaussian GEE estimating function with independence working correlation
mu.Lin <- function(eta) {
  g.Lin <- function(m) {
    v.Lin <- function(eta) (rep(1,length(eta)))
    EE.fn.ind <- function(Y,X,b) {
      ee.GEE(Y,X,b, 
      mu.Y=mu.Lin, 
      g.Y=g.Lin, 
      v.Y=v.Lin, 
      id=indiv.index, 
      corstr="ind")
    }
    ## These two give the same result
    coef.mat <- eeboost(Y,X,EE.fn.ind,maxit=250)
    coef.mat2 <- geeboost(Y,X,id=indiv.index,family="gaussian",corstr="ind",maxit=250)$coefmat
  }
  par(mfrow=c(1,2))
  coef_traceplot(coef.mat)
  coef_traceplot(coef.mat2)
Index

cof_traceplot, 2, 4

ee.GEE, 2
eeboost, 2, 3

ggeeboost, 2, 4, 6, 7
QIC, 5

scale, 6

threeboost, 2–4, 6
threeboost-package, 2