Package ‘GPLTR’

April 23, 2014

Type Package

Title Generalized Partially Linear Tree-based Regression Model

Version 0.85

Date 2014-04-22

Author Cyprien Mbogning <cyprien.mbogning@inserm.fr> and Wilson Toussile

Maintainer Cyprien Mbogning <cyprien.mbogning@gmail.com>

Description Package for generalized partially linear tree-based regression model, combining a generalized linear model with an additional tree part on the same scale.

License GPL (>=2.0)

LazyLoad yes

Depends rpart , parallel

R topics documented:

GPLTR-package *Fit a generalized partially linear tree-based regression model*

Description

Package for generalize partially linear tree-based regression model, combining a generalized linear model with an additional tree part on the same scale.

Details

Package:	GPLTR
Type:	Package
Version:	0.85
Date:	2014-04-22
License:	GPL(>=2.0)

Author(s)

Cyprien Mbogning and Wilson Toussile

Maintainer: Cyprien Mbogning <cyprien.mbogning@gmail.com>

References

Mbogning, C., Perdry, H., Toussile, W., Broet, P.: A novel tree-based procedure for deciphering the genomic spectrum of clinical disease entities. Journal of Clinical Bioinformatics 4:6, (2014)

Terry M. Therneau, Elizabeth J. Atkinson (2013) An Introduction to Recursive Partitioning Using the RPART Routines. Mayo Foundation.

Chen, J., Yu, K., Hsing, A., Therneau, T.M.: A partially linear tree-based regression model for assessing complex joint gene-gene and gene-environment effects. Genetic Epidemiology 31, 238-251 (2007)

Examples

```
##load the data set

#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

# build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                     family = family,iterMax = 4, iterMin = 3)

#plot(fit_pltr$tree, main = 'MAXIMAL TREE')
```

```
#text(fit_pltr$tree, minlength = 0L, xpd = TRUE)

## prunned back the maximal tree by BIC or AIC criterion

#tree_select <- best.tree.BIC.AIC(xtree = fit_pltr$tree,data_pltr,Y.name,
#                            X.names, G.names, family = family)

#plot(tree_select$tree$BIC, main = 'BIC TREE')
#text(tree_select$tree$BIC, minlength = 0L,xpd = TRUE)

## prunned back the maximal tree by a cross-validation procedure

#tree_selected <- best.tree.CV(fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
# family = family, args.rpart = args.rpart, epsi = 0.001, iterMax = 15,
#      iterMin = 8, ncv = 10)

#plot(tree_selected$tree, main = 'CV TREE')
#text(tree_selected$tree, minlength = 0L, xpd = TRUE)

## Compute the p-value of the selected tree by BIC

#args.parallel = list(numWorkers = 1, type = "PSOCK")
#index = tree_select$best_index[[1]]
#p_value <- p.val.tree(xtree = fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
#            B = 10, args.rpart = args.rpart, epsi = 1e-3,
#            iterMax = 15, iterMin = 8, family = family, LB = FALSE,
#            args.parallel = args.parallel, index = index)

## select an test the selected tree by a pametric bootstrap  procedure

#best_bootstrap <- best.tree.bootstrap(fit_pltr$tree, data_pltr, Y.name, X.names,
#   G.names, B = 10, BB = 10, args.rpart = args.rpart, epsi = 0.001,
#   iterMax = 15, iterMin = 8, family = family, LEVEL = 0.05,LB = FALSE,
#                  args.parallel = args.parallel)


## bagging a set of PLTR predictors

#bagging_pred <- bagging.pltr(data_pltr, Y.name, X.names, G.names, family,
#   args.rpart,epsi = 0.001, iterMax = 15, iterMin = 8, LB = FALSE,
#  args.parallel = args.parallel, Bag = 20, Pred_Data = data.frame())

##%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
## Example on a public dataset: the burn data
##%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
## The burn data are also displayed in the KMsurv package
##%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 data(burn)

## Build the rpart tree with all the variables

 cfit = rpart(D2 ~ Z1  + Z2 + Z3 + Z4 + Z5 + Z6 + Z7 + Z8 + Z9 + Z10 + Z11,
              data = burn, method = "class")

 plot(cfit, main = 'rpart tree')
 text(cfit, xpd = TRUE)
```

```
## fit the PLTR model after adjusting on gender (Z2) using the proposed method

 args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0)
 family <- "binomial"
 X.names = "Z2"
 Y.name = "D2"
 G.names = c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')
fit_pltr <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart,
           family = family,iterMax = 4, iterMin = 3, verbose = FALSE)

tree_select <- best.tree.BIC.AIC(xtree = fit_pltr$tree, burn ,Y.name,
                             X.names, G.names, family = family)

plot(tree_select$tree$BIC, main = 'new PLTR tree')
text(tree_select$tree$BIC, xpd = TRUE)
summary(tree_select$fit_glm$BIC)

## fit the PLTR model after adjusting on gender (Z2) using the original method

## uncomment the following code and set numWorkers = 1 on a windows plateform

# args.parallel = list(numWorkers = 10, type = "PSOCK")
#best_bootstrap <- best.tree.bootstrap(fit_pltr$tree, burn, Y.name, X.names,
#           G.names, B = 2000, BB = 2000, args.rpart = args.rpart, epsi = 0.008,
#              iterMax = 6,iterMin = 5, family = family, LEVEL = 0.05,LB = FALSE,
#              args.parallel = args.parallel, verbose = FALSE)
# plot(best_bootstrap$selected_model$tree, main = 'original method')
# text(best_bootstrap$selected_model$tree, xpd = TRUE)
```

bagging.pltr *bagging pltr models*

Description

bagging procedure to agregate several PLTR models for accurate prediction

Usage

```
bagging.pltr(xdata, Y.name, X.names, G.names, family = "binomial",
args.rpart,epsi = 0.001, iterMax = 15, iterMin = 8, LB = FALSE,
args.parallel = list(numWorkers = 10, type = "PSOCK"),
Bag = 20, Pred_Data = data.frame(), verbose = TRUE)
```

Arguments

`xdata`	the learning data frame
`Y.name`	the name of the binary dependent variable
`X.names`	the names of independent variables to consider in the linear part of the glm and as offset in the tree part
`G.names`	the names of independent variables to consider in the tree part of the hybrid glm.

`family`	the glm family considered depending on the type of the dependent variable (only the binomial family works in this function for the moment) .
`args.rpart`	a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
`epsi`	a treshold value to check the convergence of the algorithm
`iterMax`	the maximal number of iteration to consider
`iterMin`	the minimum number of iteration to consider
`LB`	a binary indicator with values TRUE or FALSE indicating weither the loading is balanced or not in the parallel computing. It is nevertheless useless on a windows platform. See `parallel`
`args.parallel`	a list of two elements containing the number of workers and the type of parallelization to achieve see `parallel`.
`Bag`	The number of Bagging samples to consider
`Pred_Data`	An optional data frame to validate the bagging procedure (the test dataset)
`verbose`	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

A list with six elements

`IND_OOB`	A list of length `Bag` containing the Out Of Bag (OOB) individual for each PLTR model
`OOB_ERRORS`	A vector of length `Bag` containing OOB error of each PLTR model
`OOB_ERROR`	The OOB error of the Bagging procedure
`Tree_BAG`	A list of length `Bag` containing the bagging trees
`Glm_BAG`	A list of length `Bag` containing the bagging pltr model; could be helpfull for prediction of new features.
`TEST`	A value of "000" if `Pred_Data` is not available. A list of three elements otherwise: `PRED_ERROR`: the estimated error of the Bagging procedure on the test sample; `PRED_IND`: a matrix with the prediction of the testing data individuals with each bagging PLTR model (column by column); `FINAL_PRED_IND`: A vector with the final prediction of each individual of the testing data by the bagging procedure (the modal prediction).
`Timediff`	The execution time of the bagging procedure

Author(s)

Cyprien Mbogning

Examples

```
##load the data set
#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")


## bagging a set of PLTR predictors
#args.parallel = list(numWorkers = 1, type = "PSOCK")

#bagging_pred <- bagging.pltr(data_pltr, Y.name, X.names, G.names, family,
#              args.rpart,epsi = 0.001, iterMax = 15, iterMin = 8, LB = FALSE,
#              args.parallel = args.parallel, Bag = 20, Pred_Data = data.frame())
```

---

best.tree.BIC.AIC *Prunning the Maximal tree*

Description

this function is set to prune back the maximal tree by using the BIC or the AIC criterion.

Usage

```
best.tree.BIC.AIC(xtree, xdata, Y.name, X.names,
   G.names, family = "binomial", verbose = TRUE)
```

Arguments

xtree	a tree to prune
xdata	the dataset used to build the tree
Y.name	the name of the dependent variable
X.names	the names of independent variables to consider in the linear part of the glm
G.names	the names of independent variables to consider in the tree part of the hybrid glm.
family	the glm family considered depending on the type of the dependent variable.
verbose	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

a list of four elements:

best_index	The size of the selected trees by BIC and AIC
tree	The selected trees by BIC and AIC
fit_glm	The fitted pltr models selected with BIC, and AIC
Timediff	The execution time of the selection procedure

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Toussile, W., Broet, P.: A novel tree-based procedure for deciphering the genomic spectrum of clinical disease entities. Journal of Clinical Bioinformatics 4:6, (2014)

Akaike, H.: A new look at the statistical model identification. `IEEE Trans. Automat. Control AC-19`, 716-723 (1974)

Schwarz, G.: Estimating the dimension of a model. `The Annals of Statistics` 6, 461-464 (1978)

See Also

`best.tree.CV`, `pltr.glm`

Examples

```
##load the data set
#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

## build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                      family = family,iterMax =15, iterMin = 8)

##prunned back the maximal tree by BIC or AIC criterion

#tree_select <- best.tree.BIC.AIC(xtree = fit_pltr$tree,data_pltr,Y.name,
#                             X.names, G.names, family = family)

#plot(tree_select$tree$BIC, main = 'BIC TREE')
#text(tree_select$tree$BIC, minlength = 0L, xpd = TRUE)
```

`best.tree.bootstrap` *parametric bootstrap on a pltr model*

Description

a parametric bootstrap procedure to select and test at the same time the selected tree

Usage

```
best.tree.bootstrap(xtree, xdata, Y.name, X.names, G.names, B = 10, BB = 10,
args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10), epsi = 0.001,
iterMax = 15, iterMin = 8, family = "gaussian", LEVEL = 0.05, LB = FALSE,
args.parallel = list(numWorkers = 10, type = "PSOCK"), verbose = TRUE)
```

Arguments

`xtree`	the maximal tree obtained by the function pltr.glm
`xdata`	the data frame used to build xtree
`Y.name`	the name of the dependent variable
`X.names`	the names of independent variables to consider in the linear part of the glm
`G.names`	the names of independent variables to consider in the tree part of the hybrid glm.
`B`	the size of the bootstrap sample
`BB`	the size of the bootstrap sample to compute the adjusted p-value
`args.rpart`	a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
`epsi`	a treshold value to check the convergence of the algorithm
`iterMax`	the maximal number of iteration to consider
`iterMin`	the minimum number of iteration to consider
`family`	the glm family considered depending on the type of the dependent variable.
`LEVEL`	the level of the test
`LB`	a binary indicator with values TRUE or FALSE indicating weither the loading is balanced or not in the parallel computing. It is useless on a windows platform.
`args.parallel`	a list of two elements containing the number of workers and the type of parallelization to achieve
`verbose`	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

a list with six elements

`selected_model`	a list with the fit of the selected pltr model `fit_glm`, the selected tree `tree`, the p-value of the selected tree `p.value`, the ajusted p-value of the selected tree `adj_p.value` and an indicator `Tree_Selected` to assess wether the test is significant or not.
`fit_glm`	the fitted pltr model under the null hypothesis if the test is not significant
`Timediff`	The execution time of the `parametric bootstrap` procedure
`comp_p_values`	The P-values of the competing trees
`Badj`	The number of samples used in the inner level of the procedure
`BBadj`	The number of samples used in the outer level of the procedure

Author(s)

Cyprien Mbogning

References

Chen, J., Yu, K., Hsing, A., Therneau, T.M.: A partially linear tree-based regression model for assessing complex joint gene-gene and gene-environment effects. Genetic Epidemiology 31, 238-251 (2007)

See Also

p.val.tree

Examples

```
#load the data set
data(data_pltr)
args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#             family = family,iterMax =15, iterMin = 8)

## select an test the selected tree by a parametric bootstrap procedure
#args.parallel = list(numWorkers = 1, type = "PSOCK")

#best_bootstrap <- best.tree.bootstrap(fit_pltr$tree, data_pltr, Y.name, X.names,
#  G.names, B = 10, BB = 10, args.rpart = args.rpart, epsi = 0.001,
#  iterMax = 15, iterMin = 8, family = family, LEVEL = 0.05,LB = FALSE,
#  args.parallel = args.parallel)
```

---

best.tree.CV *Prunning the Maximal tree*

Description

this function is set to prune back the maximal tree by using a K-fold `cross-validation` procedure.

Usage

```
best.tree.CV(xtree, xdata, Y.name, X.names, G.names, family = "binomial",
args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10), epsi = 0.001,
iterMax = 15, iterMin = 8, ncv = 10, verbose = TRUE)
```

Arguments

`xtree`	a tree to prune
`xdata`	the dataset used to build the tree
`Y.name`	the name of the dependent variable
`X.names`	the names of independent variables to consider in the linear part of the `glm`

G.names	the names of independent variables to consider in the tree part of the hybrid glm.
family	the glm family considered depending on the type of the dependent variable.
args.rpart	a list of options that control details of the rpart algorithm. minbucket: the minimum number of observations in any terminal <leaf> node; cp: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); maxdepth: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See rpart.control for further details
epsi	a treshold value to check the convergence of the algorithm
iterMax	the maximal number of iteration to consider
iterMin	the minimum number of iteration to consider
ncv	The number of folds to consider for the cross-validation
verbose	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

a list of five elements:

best_index	The size of the selected tree by the cross-validation procedure
tree	The selected tree by CV
fit_glm	The fitted gpltr models selected with CV
CV_ERROR	The cross-validation error of the selected tree by the CV procedure and a vector of cross-validation error of all the competing models
Timediff	The execution time of the CV procedure

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Toussile, W., Broet, P.: A novel tree-based procedure for deciphering the genomic spectrum of clinical disease entities. Journal of Clinical Bioinformatics 4:6, (2014)

See Also

best.tree.BIC.AIC, pltr.glm

Examples

```
##load the data set
#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

# build a maximal tree
```

```
#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                     family = family,iterMax =15, iterMin = 8)

##prunned back the maximal tree by a cross-validation procedure

#tree_selected <- best.tree.CV(fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
#     family = family, args.rpart = args.rpart, epsi = 0.001, iterMax = 15,
#     iterMin = 8, ncv = 10)

#plot(tree_selected$tree, main = 'CV TREE')
#text(tree_selected$tree, minlength = 0L, xpd = TRUE)
```

---

best.tree.permute *permutation test on a pltr model*

Description

a unified permutation test procedure to select and test at the same time the selected tree

Usage

```
best.tree.permute(xtree, xdata, Y.name, X.names, G.names, B = 10,
 args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10), epsi = 0.001,
 iterMax = 15, iterMin = 8, family = "binomial", LEVEL = 0.05,
 LB = FALSE, args.parallel = list(numWorkers = 10, type = "PSOCK"), verbose = TRUE)
```

Arguments

`xtree`	the maximal tree obtained by the function pltr.glm
`xdata`	the data frame used to build xtree
`Y.name`	the name of the dependent variable
`X.names`	the names of independent variables to consider in the linear part of the glm. For this function, only a binary variable is supported.
`G.names`	the names of independent variables to consider in the tree part of the hybrid glm.
`B`	the size of the bootstrap sample
`args.rpart`	a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; cp: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
`epsi`	a treshold value to check the convergence of the algorithm
`iterMax`	the maximal number of iteration to consider
`iterMin`	the minimum number of iteration to consider
`family`	the binomial family.
`LEVEL`	the level of the test
`LB`	a binary indicator with values TRUE or FALSE indicating weither the loading is balanced or not in the parallel computing. It is useless on a windows platform.

`args.parallel`	a list of two elements containing the number of workers and the type of parallelization to achieve
`verbose`	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

a list with six elements:

`p.val_selected`	the adjusted p-value of the selected tree
`selected_model`	a list with the fit of the selected pltr model `fit_glm`, the selected tree `tree` and the p-value of the selected tree without adjusting for multiple comparisons `p.value`
`fit_glm`	the fitted pltr model under the null hypothesis if the test is not significant
`Timediff`	The execution time of the `permutation test` procedure
`comp_p_values`	The P-values of the competing trees
`Badj`	The number of samples used inside the procedure

Author(s)

Cyprien Mbogning

See Also

`p.val.tree`, `best.tree.bootstrap`

Examples

```
##load the data set
#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

## build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                      family = family,iterMax =15, iterMin = 8)

## select an test the selected tree by a permutation test procedure
#args.parallel = list(numWorkers = 1, type = "PSOCK")

#best_permute <- best.tree.permute(fit_pltr$tree, data_pltr, Y.name, X.names,
#  G.names, B = 10, args.rpart = args.rpart, epsi = 0.001, iterMax = 15,
#              iterMin = 8, family = family, LEVEL = 0.05,LB = FALSE,
#                                   args.parallel = args.parallel)
```

burn — *burn dataset*

Description

The burn data frame has 154 rows and 17 columns.

Usage

```
data(burn)
```

Format

A data frame with 154 observations on the following 17 variables.

- `Obs` Observation number
- `Z1` Treatment: 0-routine bathing 1-Body cleansing
- `Z2` Gender (0=male 1=female)
- `Z3` Race: 0=nonwhite 1=white
- `Z4` Percentage of total surface area burned
- `Z5` Burn site indicator: head 1=yes, 0=no
- `Z6` Burn site indicator: buttock 1=yes, 0=no
- `Z7` Burn site indicator: trunk 1=yes, 0=no
- `Z8` Burn site indicator: upper leg 1=yes, 0=no
- `Z9` Burn site indicator: lower leg 1=yes, 0=no
- `Z10` Burn site indicator: respiratory tract 1=yes, 0=no
- `Z11` Type of burn: 1=chemical, 2=scald, 3=electric, 4=flame
- `T1` Time to excision or on study time
- `D1` Excision indicator: 1=yes 0=no
- `T2` Time to prophylactic antibiotic treatment or on study time
- `D2` Prophylactic antibiotic treatment: 1=yes 0=no
- `T3` Time to straphylocous aureaus infection or on study time
- `D3` Straphylocous aureaus infection: 1=yes 0=no

Source

Klein and Moeschberger (1997) Survival Analysis Techniques for Censored and truncated data, `Springer`.

Ichida et al. `Stat. Med.` 12 (1993): 301-310.

Examples

```
data(burn)
## maybe str(burn) ;
```

`data_pltr` *gpltr data example*

Description

A data frame to test the functions of the package

Usage

```
data(data_pltr)
```

Format

A data frame with 3000 observations on the following 16 variables.

`G1` a numeric vector

`G2` a factor with levels `0` `1`

`G3` a factor with levels `0` `1`

`G4` a factor with levels `0` `1`

`G5` a factor with levels `0` `1`

`G6` a binary numeric vector

`G7` a binary numeric vector

`G8` a binary numeric vector

`G9` a binary numeric vector

`G10` a binary numeric vector

`G11` a binary numeric vector

`G12` a binary numeric vector

`G13` a binary numeric vector

`G14` a binary numeric vector

`G15` a binary numeric vector

`Y` a binary numeric vector

Details

The numeric variable `G1` is considered as offset in the simulated `PLTR` model; the variables `G2`,...,`G5` are used to simulate the tree part, while `G6`,...,`G15` are noise variables.

Examples

```
data(data_pltr)
## maybe str(data_pltr) ...
```

nested.trees *compute the nested trees*

Description

Compute a set of nested competing trees for the prunning phase

Usage

```
nested.trees(xtree, xdata, Y.name, X.names, G.names, MaxTreeSize = NULL,
family = "gaussian", verbose = TRUE)
```

Arguments

`xtree`	a tree inheriting to the rpart method
`xdata`	the dataset used to build the tree
`Y.name`	the name of the dependent variable in the tree model
`X.names`	the names of independent variables considered as offset in the tree model
`G.names`	the names of independent variables used to build the tree.
`MaxTreeSize`	The maximal size of the competing trees
`family`	the glm family considered depending on the type of the dependent variable.
`verbose`	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

a list with 4 elements:

`leaves`	a list of leaves of the competing trees to consider for the optimal tree
`null_deviance`	the deviance of the null model (linear part of the glm)
`deviances`	a vector of deviances of the competing PLTR models
`diff_deviances`	a vector of the deviance differencies between the competing PLTR models and the null model

Author(s)

Cyprien Mbogning

Examples

```
##load the data set

#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

## build a maximal tree
```

```
#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                     family = family,iterMax =15, iterMin = 8)

## compute the competing trees
#nested_trees <- nested.trees(fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
#                             MaxTreeSize = 10, family = family)
```

p.val.tree *Compute the p-value*

Description

Test weither the selected tree by either BIC, AIC or CV procedure is significantly associated to the dependent variable or not, while adjusting for a confounding effect.

Usage

```
p.val.tree(xtree, xdata, Y.name, X.names, G.names, B = 10, args.rpart =
list(minbucket = 40, maxdepth = 10, cp = 0), epsi = 0.001, iterMax = 15,
iterMin = 8, family = "binomial", LB = FALSE,
args.parallel = list(numWorkers = 10, type = "PSOCK"), index = 4, verbose = TRUE)
```

Arguments

xtree	the maximal tree obtained by the function pltr.glm
xdata	the data frame used to build xtree
Y.name	the name of the dependent variable
X.names	the names of independent variables to consider in the linear part of the glm
G.names	the names of independent variables to consider in the tree part of the hybrid glm.
B	the resampling size of the deviance difference
args.rpart	a list of options that control details of the rpart algorithm. minbucket: the minimum number of observations in any terminal <leaf> node; cp: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); maxdepth: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See rpart.control for further details
epsi	a treshold value to check the convergence of the algorithm
iterMax	the maximal number of iteration to consider
iterMin	the minimum number of iteration to consider
family	the glm family considered depending on the type of the dependent variable.
LB	a binary indicator with values TRUE or FALSE indicating weither the loading are balanced or not in the parallel computing
args.parallel	a list of two elements containing the number of workers and the type of parallelization to achieve
index	the size of the selected tree (by the functions best.tree.BIC.AIC or best.tree.CV) using one of the proposed criteria
verbose	Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

A list of three elements:

`p.value`	The `P-value` of the selected tree
`Timediff`	The execution time of the `test` procedure
`Badj`	The number of samples used inside the the procedure

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Toussile, W., Broet, P.: A novel tree-based procedure for deciphering the genomic spectrum of clinical disease entities. Journal of Clinical Bioinformatics 4:6, (2014)

Fan, J., Zhang, C., Zhang, J.: Generalized likelihood ratio statistics and WILKS phenomenon. Annals of Statistics 29(1), 153-193 (2001)

See Also

`best.tree.bootstrap`, `best.tree.permute`

Examples

```
#load the data set

#data(data_pltr)
#args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

## build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                     family = family,iterMax =15, iterMin = 8)

##prunned back the maximal tree by BIC or AIC criterion

#tree_select <- best.tree.BIC.AIC(xtree = fit_pltr$tree,data_pltr,Y.name,
#                            X.names, G.names, family = family)

## Compute the p-value of the selected tree by BIC

#args.parallel = list(numWorkers = 1, type = "PSOCK")
#index = tree_select$best_index[[1]]
#p_value <- p.val.tree(xtree = fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
#            B = 10, args.rpart = args.rpart, epsi = 1e-3,
#            iterMax = 15, iterMin = 8, family = family, LB = FALSE,
#            args.parallel = args.parallel, index = index)
```

pltr.glm *Partially tree-based regression model function*

Description

The `pltr.glm` function is designed to fit an hybrid glm model with an additive tree part on a glm scale.

Usage

```
pltr.glm(data, Y.name, X.names, G.names, family = "binomial",
    args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10),
    epsi = 0.001, iterMax = 15, iterMin = 8, verbose = TRUE)
```

Arguments

`data`	a data frame containing the variables in the model
`Y.name`	the name of the dependent variable
`X.names`	the names of independent variables to consider in the linear part of the `glm`
`G.names`	the names of independent variables to consider in the tree part of the hybrid `glm`.
`family`	the `glm` family considered depending on the type of the dependent variable.
`args.rpart`	a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
`epsi`	a treshold value to check the convergence of the algorithm
`iterMax`	the maximal number of iteration to consider
`iterMin`	the minimum number of iteration to consider
`verbose`	Logical; TRUE for printing progress during the computation (helpful for debugging)

Details

The `pltr.glm` function use an itterative procedure to fit the linear part of the `glm` and the tree part. The tree obtained at the convergence of the procedure is a maximal tree which overfits the data. It's then mandatory to prunned back this tree by using one of the proposed criteria (`BIC`, `AIC` and `CV`).

Value

A list with four elements:

`fit`	the global glm fitted at the end of the algorithm
`tree`	the maximal tree obtained at the end of the algorithm
`nber_iter`	the number of iterations used by the algorithm
`Timediff`	The execution time of the iterative procedure

Note

The tree obtained at the end of these itterative procedure usually ovrfits the data. It's therefore mendatory to use either `best.tree.BIC.AIC` or `best.tree.CV` to prunne back the tree.

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Toussile, W., Broet, P.: A novel tree-based procedure for deciphering the genomic spectrum of clinical disease entities. Journal of Clinical Bioinformatics 4:6, (2014)

Terry M. Therneau, Elizabeth J. Atkinson (2013) An Introduction to Recursive Partitioning Using the RPART Routines. Mayo Foundation.

Chen, J., Yu, K., Hsing, A., Therneau, T.M.: A partially linear tree-based regression model for assessing complex joint gene-gene and gene-environment effects. Genetic Epidemiology 31, 238-251 (2007)

See Also

`rpart`

Examples

```
##load the data set

data(data_pltr)
args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree (uncomment the following code)

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                    family = family,iterMax =15, iterMin = 8)

#plot(fit_pltr$tree, main = 'MAXIMAL TREE')
#text(fit_pltr$tree, minlength = 0L, xpd = TRUE)
```

`predict_bagg.pltr` *prediction on new features*

Description

Predict new features using a set of bagging pltr models

Usage

```
predict_bagg.pltr(bag_pltr, Y.name, newdata, type = "response",
                  thresshold = 0.5)
```

Arguments

`bag_pltr`	a list containing the set of bagging pltr model obtained with the function `bagging.pltr`
`Y.name`	the name of the binary dependent variable
`newdata`	a data frame in which to look for variables with which to predict.
`type`	the type of prediction required. `type = "response"` is the default; It gives the predicted probabilities. At this stage of the package, only this type is take into account. Other types like `"link"` and `"terms"` aren't supported yet.
`thresshold`	the cutoff value for binary prediction

Value

A list with three elements

`PRED_IND`	A vector with the final prediction of each individual of the testing data by the bagging procedure (the modal prediction).
`PRED_IND`	a matrix with the prediction of the testing data individuals with each bagging PLTR model (column by column)
`PRED_ERROR`	the estimated error of the Bagging procedure on the test sample.

Author(s)

Cyprien Mbogning

See Also

`bagging.pltr`, `predict.glm`

Examples

```
##
```

`tree2glm` *tree to GLM*

Description

fit the PLTR model for a given tree. The tree is tranform in indicators covariates.

Usage

```
tree2glm(xtree, xdata, Y.name, X.names, G.names, family = "gaussian")
```

Arguments

`xtree`	a tree inherits from the rpart method
`xdata`	a data frame containing the variables in the model
`Y.name`	the name of the dependent variable
`X.names`	the names of independent variables to consider in the linear part of the glm
`G.names`	the names of independent variables considered in the tree part.
`family`	the glm family considered depending on the type of the dependent variable.

Value

the fitted model (fit)

Author(s)

Cyprien Mbogning

Examples

```
##load the data set

#data(data_pltr)
#args.rpart <- list(minbucket = 40, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

## build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                     family = family,iterMax = 15, iterMin = 8)

#fit_glm <- tree2glm(fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
#                    family = family)

#summary(fit_glm)
```

`tree2indicators` *From a tree to indicators*

Description

Coerces a given tree structure to binary covariates.

Usage

```
tree2indicators(fit)
```

Arguments

`fit`	a tree structure inheriting to the rpart method

Value

a list of indicators

Author(s)

Cyprien Mbogning

Examples

```
##load the data set

#data(data_pltr)
#args.rpart <- list(minbucket = 40, xval = 10, cp = 0)
#family <- "binomial"
#Y.name <- "Y"
#X.names <- "G1"
#G.names <- paste("G", 2:15, sep="")

## build a maximal tree

#fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
#                     family = family,iterMax = 15, iterMin = 8)

##Compute a list of indicator from the leaves of the tree fitted tree

#tree2indicators(fit_pltr$tree)
```

Index