Package 'GPLTR'

December 19, 2014

Type Package

Title Generalized Partially Linear Tree-based Regression Model

Version 1.1

Date 2014-12-19

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

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

Description Combining a generalized linear model with an additional tree part
on the same scale. A four-step procedure is proposed to fit the model and test
the joint effect of the selected tree part while adjusting on confounding factors.
We also proposed an ensemble procedure based on the bagging to improve prediction
accuracy and computed several scores of importance for variable selection.

License GPL (>=2.0)

LazyLoad yes

Depends rpart , parallel

R topics documented:

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

Description

Combining a generalized linear model with an additional tree part on the same scale. A four-step procedure is proposed to fit the model and test the joint effect of the selected tree part while adjusting on confounding factors. We also proposed an ensemble procedure based on the bagging to improve prediction accuracy and computed several scores of importance for variable selection.

Details

Package:	GPLTR
Type:	Package
Version:	1.1
Date:	2014-12-19
License:	GPL(>=2.0)

Author(s)

Cyprien Mbogning and Wilson Toussile

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

References

Mbogning, C., Perdry, H., Broet, P.: A Bagged partially linear tree-based regression procedure for prediction and variable selection (submitted 2014)

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

```
##%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
## Example on a public dataset: the burn data
##%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
## The burn data are also displayed in the KMsurv package
##%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 ## Not run:
 data(burn)

## Build the rpart tree with all the variables
```

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

 plot(rpart.burn, main = 'rpart tree')
 text(rpart.burn, xpd = TRUE, cex = .6, use.n = TRUE)

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

 args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxcompete = 0,
                    maxsurrogate = 0)
 family <- "binomial"
 X.names = "Z2"
 Y.name = "D2"
 G.names = c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')

pltr.burn <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart,
                   family = family, iterMax = 4, iterMin = 3, verbose = FALSE)

## Prunned back the maximal tree using either the BIC or the AIC criterion

pltr.burn_prun <- best.tree.BIC.AIC(xtree = pltr.burn$tree, burn, Y.name,
                                    X.names, family = family)

## plot the BIC selected tree

plot(pltr.burn_prun$tree$BIC, main = 'BIC selected tree')
text(pltr.burn_prun$tree$BIC, xpd = TRUE, cex = .6, col = 'blue')

## Summary of the selected tree by a BIC criterion

summary(pltr.burn_prun$tree$BIC)

## Summary of the final selected pltr model

summary(pltr.burn_prun$fit_glm$BIC)

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

##  set numWorkers = 1 on a windows plateform

 args.parallel = list(numWorkers = 10)

best_bootstrap <- best.tree.bootstrap(pltr.burn$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 a set of basic unprunned pltr predictors
# ?bagging.pltr

Bag.burn <-  bagging.pltr(burn, Y.name, X.names, G.names, family,
              args.rpart,epsi = 0.01, iterMax = 4, iterMin = 3,
```

```
              Bag = 10, verbose = FALSE, doprune = FALSE)

## The thresshold values used

Bag.burn$CUT

## The set of PLTR models in the bagging procedure

PLTR_BAG.burn <- Bag.burn$Glm_BAG

## The set of trees in the bagging procedure

TREE_BAG.burn <- Bag.burn$Tree_BAG

## Use the bagging procedure to predict new features
# ?predict_bagg.pltr

Pred_Bag.burn <- predict_bagg.pltr(Bag.burn, Y.name, newdata = burn,
                 type = "response", thresshold = seq(0, 1, by = 0.1))

## The confusion matrix for each thresshold value using the majority vote

Pred_Bag.burn$CONF1

## The prediction error for each thresshold value

 Pred_Bag.burn$PRED_ERROR1

## Compute the variable importances using the bagging procedure

 Var_Imp_BAG.burn <- VIMPBAG(Bag.burn, burn, Y.name)

## Importance score using the permutaion method for each thresshold value

Var_Imp_BAG.burn$PIS

## Shadow plot of three proposed scores

par(mfrow=c(1,3))
barplot(Var_Imp_BAG.burn$PIS$CUT5, main = 'PIS', horiz = TRUE, las = 1,
        cex.names = .8, col = 'lightblue')
barplot(Var_Imp_BAG.burn$DIS, main = 'DIS', horiz = TRUE, las = 1,
        cex.names = .8, col = 'grey')
barplot(Var_Imp_BAG.burn$DDIS, main = 'DDIS', horiz = TRUE, las = 1,
        cex.names = .8, col = 'purple')

## End(Not run)
```

---

bag.aucoob *AUC on the Out Of Bag samples*

Description

Compute the AUC on the OOB samples of the bagging procedure for the binomial family. The true and false positive rates are also returned and could be helpfull for plotting the ROC curves.

Usage

```
bag.aucoob(bag_pltr, xdata, Y.name)
```

Arguments

bag_pltr	The output of the function bagging.pltr
xdata	The learning dataset containing the dependent variable, the confounding variables and the predictors variables
Y.name	The name of the binary dependent variable

Details

The thresshold values used for computing the AUC are defined when building the bagging predictor. see bagging.pltr for the convenient parameterization.

Value

A list of 4 elements

AUCOOB	the AUC computed on OOB samples of the Bagging procedure
TPR	the true positive rate for several thresshold values
FPR	the false positive rate for several thresshold values
OOB	the Out Of Bag error for each thresshold value

Note

The plot of the ROC curve is straighforward using the TPR and FPR obtained with the function bag.aucoob

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Broet, P.: A Bagged partially linear tree-based regression procedure for prediction and variable selection (submitted 2014)

Examples

```
##
```

bagging.pltr *bagging pltr models*

Description

bagging procedure to agregate several PLTR models for accurate prediction and variable selection

Usage

```
bagging.pltr(xdata, Y.name, X.names, G.names, family = "binomial",
args.rpart,epsi = 0.001, iterMax = 5, iterMin = 3, LB = FALSE,
args.parallel = list(numWorkers = 1),
Bag = 20, Pred_Data = data.frame(), verbose = TRUE, doprune = FALSE
, thresshold = seq(0, 1, by = 0.1))
```

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 mclapply
args.parallel	a list of two elements containing the number of workers and the type of parallelization to achieve see mclapply.
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)
doprune	a binary indicator with values TRUE or FALSE indicating weither the set of trees in the bagging procedure are pruned (by a BIC procedure) or not
thresshold	a vector of numerical values between 0 and 1 used as thresshold values for the computation of the OOB error rate

Details

For the Bagging procedure, it is mendatory to set `maxcompete = 0` and `maxsurrogate = 0` within the rpart arguments. This will ensured the correct calculation of the importance of variables.

Value

A list with eleven elements

`IND_OOB`	A list of length `Bag` containing the Out Of Bag (OOB) individuals for each PLTR model.
`EOOB`	The vector of OOB errors of the bagging procedure for each thresshold value.
`OOB_ERRORS_PBP`	A matrix with `Bag` columns and `thresshold` rows containing OOB error of each PLTR model in the bagging sequence for each thresshold value.
`OOB_ERROR_PBP`	A vector containing the mean of `OOB_ERRORS_PBP` for each thresshold value.
`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 on new features.
`LOST`	The 0, 1 lost matrix for OOB observations at each thresshold value
`TEST`	A value of `NULL` 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 for each thresshold value; `PRED_IND`: A list of length the length of the thresshold vector, each element containing a matrix with the prediction of the testing data individuals using each PLTR model of the bagging sequence (column by column); `FINAL_PRED_IND`: A list containing the final prediction of each individual of the testing data by the bagging procedure (the modal prediction) for each thresshold value.
`Var_IMP`	A numeric vector containing the relative variable importance of the bagging procedure
`Timediff`	The execution time of the bagging procedure
`CUT`	The thresshold value used inside the bagging procedure

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Broet, P.: A Bagged partially linear tree-based regression procedure for prediction and variable selection (submitted 2014)

Leo Breiman: Bagging Predictors. Machine Learning, 24, 123-140 (1996)

See Also

`predict_bagg.pltr`

Examples

```
## Not run:
##load the data set

data(burn)

## set the parameters

args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxsurrogate = 0)
family <- "binomial"
Y.name <- "D2"
X.names <- "Z2"
G.names <- c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')
args.parallel = list(numWorkers = 1)

## Bagging a set of basic unprunned pltr predictors

Bag.burn <-  bagging.pltr(burn, Y.name, X.names, G.names, family,
             args.rpart,epsi = 0.01, iterMax = 4, iterMin = 3,
             Bag = 20, verbose = FALSE, doprune = FALSE)

## End(Not run)
```

---

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,
                  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 confounding variables to consider in the linear part of the 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 and Wilson Toussile

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

```
data(burn)

args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxcompete = 0,
                   maxsurrogate = 0)
 family <- "binomial"
 X.names = "Z2"
 Y.name = "D2"
 G.names = c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')

pltr.burn <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart,
                  family = family, iterMax = 4, iterMin = 3, verbose = FALSE)

## Prunned back the maximal tree using either the BIC or the AIC criterion

pltr.burn_prun <- best.tree.BIC.AIC(xtree = pltr.burn$tree, burn, Y.name,
                                    X.names, family = family)

## plot the BIC selected tree

plot(pltr.burn_prun$tree$BIC, main = 'BIC selected tree')
text(pltr.burn_prun$tree$BIC, xpd = TRUE, cex = .6, col = 'blue')

## Not run:
##load the data set

data(data_pltr)
```

```
## Set the parameters

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 = 5, iterMin = 3)

##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, family = family)

plot(tree_select$tree$BIC, main = 'BIC TREE')
text(tree_select$tree$BIC, minlength = 0L, xpd = TRUE, cex = .6)


## End(Not run)
```

`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 = 5, iterMin = 3, family = "binomial", LEVEL = 0.05, LB = FALSE,
args.parallel = list(numWorkers = 1), 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	parameters of the parallelization. See mclapply for more details
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 and Wilson Toussile

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="")
## Not run:
## build a maximal tree
```

```
fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names,
   args.rpart = args.rpart, family = family, iterMax = 5, iterMin = 3)

## 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 = 5, iterMin = 3, family = family, LEVEL = 0.05,LB = FALSE,
  args.parallel = args.parallel)

## End(Not run)
```

`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 = 5, iterMin = 3, 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_ERRORS	A list of two elements containing the cross-validation error of the selected tree by the CV procedure and a vector of cross-validation errors of all the competing models
Timediff	The execution time of the Cross-Validation 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

```
## Not run:
##load the data set

data(data_pltr)

## set the parameters

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 = 5, iterMin = 3)

##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 = 5,
     iterMin = 3, ncv = 10)

plot(tree_selected$tree, main = 'CV TREE')
text(tree_selected$tree, minlength = 0L, xpd = TRUE, cex = .6)

## End(Not run)
```

`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 = 5, iterMin = 3, family = "binomial", LEVEL = 0.05,
 LB = FALSE, args.parallel = list(numWorkers = 1, 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`	parameters of the parallelization. See `mclapply` for more details.
`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

```
## Not run:
##load the data set

data(data_pltr)

## set the parameters

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 = 5, iterMin = 3)

## 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 = 5,
             iterMin = 3, family = family, LEVEL = 0.05,LB = FALSE,

## End(Not run)                                    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 sequence of nested competing trees for the prunning step

Usage

```
nested.trees(xtree, xdata, Y.name, X.names, MaxTreeSize = NULL,
family = "binomial", 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
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 and Wilson Toussile

Examples

```
## Not run:
## 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 = 5, iterMin = 3)

## compute the competing trees

nested_trees <- nested.trees(fit_pltr$tree, data_pltr, Y.name, X.names,
                             MaxTreeSize = 10, family = family)

## End(Not run)
```

`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 = 5,
iterMin = 3, family = "binomial", LB = FALSE,
args.parallel = list(numWorkers = 1), 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 confounding 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`	parameters of the parallelization. See `mclapply` for more details.

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

```
## Not run:
## load the data set

data(data_pltr)

## set the parameters

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 = 5, iterMin = 3)

##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, family = family)

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

```
args.parallel = list(numWorkers = 10, 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 = 100, args.rpart = args.rpart, epsi = 1e-3,
            iterMax = 5, iterMin = 3, family = family, LB = FALSE,
            args.parallel = args.parallel, index = index)

## End(Not run)
```

`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 = 5, iterMin = 3, 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 glm fitted on the confounding factors at the end of the iterative 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 overfits 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 and Wilson Toussile

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

```
data(burn)

args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxcompete = 0,
                   maxsurrogate = 0)
 family <- "binomial"
 X.names = "Z2"
 Y.name = "D2"
 G.names = c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')

pltr.burn <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart,
                  family = family, iterMax = 4, iterMin = 3, verbose = FALSE)


## Not run:
## load the data set

data(data_pltr)

## set the parameters

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 = 5, iterMin = 3)

plot(fit_pltr$tree, main = 'MAXIMAL TREE')
text(fit_pltr$tree, minlength = 0L, xpd = TRUE, cex = .6)

## End(Not run)
```

`predict_bagg.pltr` *prediction on new features*

Description

Prediction on new features using a set of bagging pltr models

Usage

```
predict_bagg.pltr(bag_pltr, Y.name, newdata, type = "response",
                  thresshold = seq(0, 1, by = 0.1))
```

Arguments

`bag_pltr`	the bagging result obtained with the function `bagging.pltr`
`Y.name`	the name of the binary dependent variable
`newdata`	a data frame in which to look for predictors and the dependant variable.
`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 such as `"link"` and `"terms"` are useless.
`thresshold`	a vector of cutoff values for binary prediction. Could be helpfull for computing the AUC on the test sample.

Value

A list with 8 elements

`FINAL_PRED_IND1`
: A list of size the length of the thresshold vector, containing the final prediction of each individual of the testing data by the bagging procedure using the majority rule (the modal prediction).

`FINAL_PRED_IND2`
: A list of size the length of the thresshold vector, containing the final prediction of each individual of the testing data by the bagging procedure using the mean estimated probability.

`PRED_ERROR1`
: A vector of estimated errors of the Bagging procedure on the test sample for each thresshold value using `FINAL_PRED_IND1`.

PRED_ERROR2	A vector of estimated errors of the Bagging procedure on the test sample for each thresshold value using FINAL_PRED_IND2.
CONF1	A list of confusion matrix using FINAL_PRED_IND1
CONF2	A list of confusion matrix using FINAL_PRED_IND2
PRED_ERRORS_PBP	A list of size the length of the thresshold vector. Each element representing the prediction error obtained via each predictor in the bagging sequence for each thresshold value
PRED_ERROR_PBP	A vector containing the mean of PRED_ERRORS_PBP for each thresshold value

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Broet, P.: A Bagged partially linear tree-based regression procedure for prediction and variable selection (submitted 2014)

See Also

bagging.pltr, predict.glm

Examples

```
## Not run:
## load the data set

 data(burn)

## set the parameters

 args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxsurrogate = 0)
 family <- "binomial"
 Y.name <- "D2"
 X.names <- "Z2"
 G.names <- c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')
 args.parallel = list(numWorkers = 1)

## Bagging a set of basic unprunned pltr predictors

 Bag.burn <-  bagging.pltr(burn, Y.name, X.names, G.names, family,
             args.rpart,epsi = 0.01, iterMax = 4, iterMin = 3,
             Bag = 20, verbose = FALSE, doprune = FALSE)

## Use the bagging procedure to predict new features

# ?predict_bagg.pltr

 Pred_Bag.burn <- predict_bagg.pltr(Bag.burn, Y.name, newdata = burn,
                 type = "response", thresshold = seq(0, 1, by = 0.1))

## The confusion matrix for each thresshold value using the majority vote

Pred_Bag.burn$CONF1
```

```
## End(Not run)
```

`predict_pltr` *prediction*

Description

prediction on new features using a pltr tree and the name of the confounding variable

Usage

```
predict_pltr(xtree, xdata, Y.name, X.names, newdata, type = "response",
              family = 'binomial', thresshold = seq(0.1, 0.9, by = 0.1))
```

Arguments

`xtree`	a tree obtained with the pltr procedure
`xdata`	the dataframe used to learn the pltr model
`Y.name`	the name of the main variable
`X.names`	the names of the confounding variables
`newdata`	the new data with all the predictors and the dependent variable
`type`	the type of prediction
`family`	the glm family considered
`thresshold`	the thresshold value to consider for binary prediction. It could be a vector, helping to compute the AUC

Value

A list of two element

`predict_glm`	the predicted vector, depending on the family used. For the binomial family with a vector of thresshold, a matrix with each column corresponding to a thresshold value
`ERR_PRED`	either the prediction error of the pltr procedure on the test set or a vector of prediction error when the family is binomial with a vector of thresshold values

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

`pltr.glm`, `predict.glm`

Examples

```
##
```

tree2glm *tree to GLM*

Description

fit the PLTR model for a given tree. The tree is coerced into dummy covariates.

Usage

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

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
family	the glm family considered depending on the type of the dependent variable.

Value

the pltr fitted model (fit)

Author(s)

Cyprien Mbogning and Wilson Toussile

Examples

```
## Not run:
##load the data set

data(data_pltr)

## set the parameters

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 = 5, iterMin = 3)

## Coerce a tree into a glm model using the confonding factor

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

```
summary(fit_glm)

## End(Not run)
```

`tree2indicators` *From a tree to indicators (or dummy variables)*

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 and Wilson Toussile

Examples

```
## Not run:
## load the data set

data(data_pltr)

## set the parameters

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 = 5, iterMin = 3)

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

tree2indicators(fit_pltr$tree)

## End(Not run)
```

VIMPBAG *score of importance for variables*

Description

Several variable importance scores are computed: the deviance importance score (DIS), the permutation importance score (PIS), the depth deviance importance score (DDIS), the minimal depth importance score (MinDepth) and the occurence score (OCCUR).

Usage

```
VIMPBAG(BAGGRES, data, Y.name)
```

Arguments

BAGGRES	The output of the bagging procedure (bagging.pltr)
data	The learning dataframe used within the bagging procedure
Y.name	The name of the binary dependant variable used in the bagging procedure

Details

several choices for variable selection using the bagging procedure are proposed. A discussion about the scores of importance PIS, DIS, and DDIS is available in Mbogning et al. 2014

Value

A list with 9 elements

PIS	A list of length the length of the thresshold value used in the bagging procedure, containing the permutation importance score displayed in decreasing order for each thresshold value
StdPIS	The standard error of the PIS
OCCUR	The occurence number for each variable in the bagging sequence displayed in decreasing order
DIS	The deviance importance score displayed in decreasing order
DDIS	The depth deviance importance score displayed in decreasing order
MinDepth	The minimal depth score for each variable, displayed in increasing order
dimtrees	A vector containing the dimensions of trees within the baging sequence
EOOB	A vector containing the OOB error of the bagging procedure for each thresshold value
Bagfinal	The number of Bagging iterations used

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Broet, P.: A Bagged partially linear tree-based regression procedure for prediction and variable selection (submitted 2014)

See Also

bagging.pltr

Examples

```
## Not run:
## load the data set

 data(burn)

## set the parameters

 args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxsurrogate = 0)
 family <- "binomial"
 Y.name <- "D2"
 X.names <- "Z2"
 G.names <- c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')
 args.parallel = list(numWorkers = 1)

## Bagging a set of basic unprunned pltr predictors

 Bag.burn <-  bagging.pltr(burn, Y.name, X.names, G.names, family,
             args.rpart,epsi = 0.01, iterMax = 4, iterMin = 3,
             Bag = 20, verbose = FALSE, doprune = FALSE)

## Several importance scores for variables, using the bagging procedure

 Var_Imp_BAG.burn <- VIMPBAG(Bag.burn, burn, Y.name)

## Importance score using the permutaion method for each thresshold value

 Var_Imp_BAG.burn$PIS

## Importance score using the deviance criterion

 Var_Imp_BAG.burn$DIS

## End(Not run)
```

Index