Package ‘trimTrees’

February 20, 2015

Type Package

Title Trimmed opinion pools of trees in a random forest

Version 1.2

Date 2014-08-1

Depends R (>= 2.5.0),stats,randomForest,mlbench

Author Yael Grushka-Cockayne, Victor Richmond R. Jose, Kenneth C. Lichtendahl Jr. and Huanghui Zeng, based on the source code from the randomForest package by Andy Liaw and Matthew Wiener and on the original Fortran code by Leo Breiman and Adele Cutler.

Maintainer Yael Grushka-Cockayne <grushkay@darden.virginia.edu>

Description Creates point and probability forecasts from the trees in a random forest using a trimmed opinion pool.

Suggests MASS

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-14 07:11:17

R topics documented:

`cinbag` *Modified Classification and Regression with Random Forest*

Description

cinbag implements a modified random forest algorithm (based on the source code from the randomForest package by Andy Liaw and Matthew Wiener and on the original Fortran code by Leo Breiman and Adele Cutler) to return the number of times a row appears in a tree's bag. cinbag returns a randomForest object, e.g., rfobj, with an additional output, a matrix with inbag counts (rows) for each tree (columns). For instance, rfobj$inbagCount is similar to rfobj$inbag, but with inbag counts instead of inbag indicators.

Usage

```
cinbag(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
       mtry=if (!is.null(y) && !is.factor(y))
       max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
       replace=TRUE, classwt=NULL, cutoff, strata,
       sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
       nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
       maxnodes = NULL,
       importance=FALSE, localImp=FALSE, nPerm=1,
       proximity, oob.prox=proximity,
       norm.votes=TRUE, do.trace=FALSE,
       keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
       keep.inbag=FALSE, ...)
```

Arguments

x	a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the print method, an randomForest object).
y	A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, randomForest will run in unsupervised mode.
xtest	a data frame or matrix (like x) containing predictors for the test set.
ytest	response for the test set.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3).
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one. Ignored for regression.
cutoff	(Classification only) A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).

`strata`	A (factor) variable that is used for stratified sampling.
`sampsize`	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
`nodesize`	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
`maxnodes`	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by `nodesize`). If set larger than maximum possible, a warning is issued.
`importance`	Should importance of predictors be assessed?
`localImp`	Should casewise importance measure be computed? (Setting this to `TRUE` will override `importance`.)
`nPerm`	Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.
`proximity`	Should proximity measure among the rows be calculated?
`oob.prox`	Should proximity be calculated only on “out-of-bag” data?
`norm.votes`	If `TRUE` (default), the final result of votes are expressed as fractions. If `FALSE`, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
`do.trace`	If set to `TRUE`, give a more verbose output as `randomForest` is run. If set to some integer, then running output is printed for every `do.trace` trees.
`keep.forest`	If set to `FALSE`, the forest will not be retained in the output object. If `xtest` is given, defaults to `FALSE`.
`corr.bias`	perform bias correction for regression? Note: Experimental. Use at your own risk.
`keep.inbag`	Should an `n` by `ntree` matrix be returned that keeps track of which samples are “in-bag” in which trees (but not how many times, if sampling with replacement)
`...`	optional parameters to be passed to the low level function `cinbag.default`.

Value

An object of class `randomForest`, which is a list with the following components:

`call`	the original call to `randomForest`
`type`	one of `regression`, `classification`, or `unsupervised`.
`predicted`	the predicted values of the input data based on out-of-bag samples.
`importance`	a matrix with `nclass` + 2 (for classification) or two (for regression) columns. For classification, the first `nclass` columns are the class-specific measures computed as mean descrease in accuracy. The `nclass` + 1st column is the mean descrease in accuracy over all classes. The last column is the mean decrease in Gini index. For Regression, the first column is the mean decrease in accuracy and the second the mean decrease in MSE. If `importance=FALSE`, the last measure is still returned as a vector.

`importanceSD`	The "standard errors" of the permutation-based importance measure. For classification, a `p` by `nclass + 1` matrix corresponding to the first `nclass + 1` columns of the importance matrix. For regression, a length `p` vector.
`localImp`	a `p` by `n` matrix containing the casewise importance measures, the [i,j] element of which is the importance of i-th variable on the j-th case. `NULL` if `localImp=FALSE`.
`ntree`	number of trees grown.
`mtry`	number of predictors sampled for spliting at each node.
`forest`	(a list that contains the entire forest; `NULL` if `randomForest` is run in unsupervised mode or if `keep.forest=FALSE`.
`err.rate`	(classification only) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th.
`confusion`	(classification only) the confusion matrix of the prediction (based on OOB data).
`votes`	(classification only) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) 'votes' from the random forest.
`oob.times`	number of times cases are 'out-of-bag' (and thus used in computing OOB error estimate)
`proximity`	if `proximity=TRUE` when `randomForest` is called, a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes).
`mse`	(regression only) vector of mean square errors: sum of squared residuals divided by `n`.
`rsq`	(regression only) "pseudo R-squared": 1 - `mse` / Var(y).
`test`	if test set is given (through the `xtest` or additionally `ytest` arguments), this component is a list which contains the corresponding `predicted`, `err.rate`, `confusion`, `votes` (for classification) or `predicted`, `mse` and `rsq` (for regression) for the test set. If `proximity=TRUE`, there is also a component, `proximity`, which contains the proximity among the test set as well as proximity between test and training data.
`inbag`	An indicator (1 or 0) for each training set row and each tree. The indicator is 1 if the training set row is in the tree's bag and is 0 otherwise. Note that this value is not listed in the original `randomForest` function's output, although it is implemented.
`inbagCount`	A count for each training set row and each tree. The count is the number of times the training set row is in the tree's bag. This output is not available in the original `randomForest` package. The purpose of the `cinbag` function is to augment the `randomForest` function so that it returns inbag counts. These counts are necessary for computing and ensembling the trees' empirical cumulative distribution functions.

Note

`cinbag`'s source files call the C functions `classRFmod.c` and `regRFmod.c`, which are slightly modified versions of the `randomForest`'s source files `classRF.c` and `regRF.c`, respectively.

Author(s)

Yael Grushka-Cockayne, Victor Richmond R. Jose, Kenneth C. Lichtendahl Jr. and Huanghui Zeng, based on the source code from the randomForest package by Andy Liaw and Matthew Wiener and on the original Fortran code by Leo Breiman and Adele Cutler.

References

Breiman L (2001). Random forests. Machine Learning 45 5-32.

Breiman L (2002). Manual on setting up, using, and understanding random forests V3.1. http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf.

See Also

`trimTrees`, `hitRate`

Examples

```
# Load the data
set.seed(201) # Can be removed; useful for replication
data <- as.data.frame(mlbench.friedman1(500, sd=1))
summary(data)

# Prepare data for trimming
train <- data[1:400, ]
test <- data[401:500, ]
xtrain <- train[,-11]
ytrain <- train[,11]
xtest <- test[,-11]
ytest <- test[,11]

# Run cinbag
set.seed(201) # Can be removed; useful for replication
rf <- cinbag(xtrain, ytrain, ntree=500, nodesize=5, mtry=3, keep.inbag=TRUE)
rf$inbag[,1] # First tree's inbag indicators
rf$inbagCount[,1] # First tree's inbag counts
```

`hitRate` *Empirical Hit Rates for a Crowd of Forecasters*

Description

This function calculates the empirical hit rates for a crowd of forecasters over a testing set. The function takes as its arguments the forecasters' probability integral transform (PIT) values – one for each testing set row – and the prediction interval of interest.

Usage

```
hitRate(matrixPIT, interval = c(0.25, 0.75))
```

Arguments

`matrixPIT`	A `ntest`-by-`nForecaster` matrix of PIT values where `ntest` is the number of rows in the testing set and `nForecaster` is the number of forecasters. Each column represents a different forecaster's PITs for the testing set. A PIT value is the forecaster's cdf evaluated at the realization of the response in the testing set.
`interval`	Prediction interval of interest. The default `interval=c(0.25, 0.75)` is the central 50% prediction interval.

Value

`HR`	An `nForecaster` vector of empirical hit rates – one for each forecaster. A forecaster's empirical hit rate is the percentage of PIT values that fall within [`interval[1]`,`interval[2]`], e.g., [0.25,0.75] according to the default.

Author(s)

Yael Grushka-Cockayne, Victor Richmond R. Jose, Kenneth C. Lichtendahl Jr., and Huanghui Zeng.

References

Grushka-Cockayne Y, Jose VRR, Lichtendahl KC Jr. (2014). Ensembles of overfit and overconfident forecasts, working paper.

See Also

`trimTrees`, `cinbag`

Examples

```
# Load the data
set.seed(201) # Can be removed; useful for replication
data <- as.data.frame(mlbench.friedman1(500, sd=1))
summary(data)

# Prepare data for trimming
train <- data[1:400, ]
test <- data[401:500, ]
xtrain <- train[,-11]
ytrain <- train[,11]
xtest <- test[,-11]
ytest <- test[,11]

# Run trimTrees
set.seed(201) # Can be removed; useful for replication
tt <- trimTrees(xtrain, ytrain, xtest, ytest, trim=0.15)

# Outputs from trimTrees
mean(hitRate(tt$treePITs))
```

```
hitRate(tt$trimmedEnsemblePITs)
hitRate(tt$untrimmedEnsemblePITs)
```

`trimTrees` *Trimmed Opinion Pools of Trees in Random Forest*

Description

This function creates point and probability forecasts from the trees in a random forest using Jose et al.'s trimmed opinion pool, a trimmed average of the trees' empirical cumulative distribution functions (cdf). For tuning purposes, the user can input the trimming level used in this trimmed average and then compare the scores of the trimmed and untrimmed opinion pools, or ensembles.

Usage

```
trimTrees(xtrain, ytrain, xtest, ytest=NULL, ntree = 500,
          mtry = if (!is.null(ytrain) && !is.factor(ytrain))
          max(floor(ncol(xtrain)/3), 1) else floor(sqrt(ncol(xtrain))),
          nodesize = if (!is.null(ytrain) && !is.factor(ytrain)) 5 else 1,
          trim = 0,trimIsExterior = TRUE,
          uQuantiles = seq(0.05, 0.95, 0.05), methodIsCDF = TRUE)
```

Arguments

`xtrain`	A data frame or a matrix of predictors for the training set.
`ytrain`	A response vector for the training set. If a factor, classification is assumed, otherwise regression is assumed.
`xtest`	A data frame or a matrix of predictors for the testing set.
`ytest`	A response vector for the testing set. If no testing set is passed, probability integral transform (PIT) values and scores will be returned as `NA`s.
`ntree`	Number of trees to grow.
`mtry`	Number of variables randomly sampled as candidates at each split.
`nodesize`	Minimum size of terminal nodes.
`trim`	The trimming level used in the trimmed average of the trees' empirical cdfs. For the cdf approach, the trimming level is the fraction of cdfs values to be trimmed from each end of the ordered vector of cdf values (for each support point) before the average is computed. For the moment approach, the trees' means are computed, ordered, and trimmed. The trimmed opinion pool using the moment approach is an average of the remaining trees.
`trimIsExterior`	If `TRUE`, the trimming is done exteriorly, or from the ends of the ordered vector. If `FALSE`, the trimming is done interiorly, or from the middle of the ordered vector.
`uQuantiles`	A vector of probabilities in a strictly increasing order and between 0 and 1. For instance, if `uQuantiles=c(0.25,0.75)`, then the 0.25-quantile and the 0.75-quantile of the trimmed and untrimmed ensembles are scored.
`methodIsCDF`	If `TRUE`, the method for forming the trimmed opinion pool is according to the cdf approach in Jose et al (2014). If `FALSE`, the moment approach is used.

Value

An object of class `trimTrees`, which is a list with the following components:

`forestSupport`	Possible points of support for the trees and ensembles.
`treeValues`	For the last testing set row, this component outputs each tree's `ytrain` values (not necessarily unique) that are both inbag and in the `xtest`'s terminal node. Note that the `ytrain` values may not be unique. This component is an `ntrain`-by-`ntree` matrix where `ntrain` is the number of rows in the training set.
`treeCounts`	For the last testing set row, each tree's counts of `treeValues` and lists them by their unique values. This component is an `nSupport`-by-`ntree` matrix. `nSupport` is the number of unique `ytrain` values, or support points of the forest.
`treeCumCounts`	Cumulative tally of `treeCounts` of dimension `nSupport+1`-by-`ntree`.
`treeCDFs`	Each tree's empirical cdf based on `treeCumCounts` for the last testing set row only. This component is an `nSupport+1`-by-`ntree` matrix. Note that the first row in this matrix is all zeros.
`treePMFs`	Each tree's empirical probability mass function (pmf) for the last testing set row. This component is an `nSupport`-by-`ntree` matrix.
`treeMeans`	For each testing set row, each tree's mean according to its empirical pmf. This component is an `ntest`-by-`ntree` matrix where `ntest` is the number of rows in the testing set.
`treeVars`	For each testing set row, each tree's variance according to its empirical pmf. This component is an `ntest`-by-`ntree` matrix.
`treePITs`	For each testing set row, each tree's probability integral transform (PIT), the empirical cdf evaluated at the realized `ytest` value. This component is an `ntest`-by-`ntree` matrix. If `ytest` is `NULL`, `NA`s are returned.
`treeQuantiles`	For the last testing set row, each tree's quantiles – one for each element in `uQuantiles`, the empirical cdf evaluated at the realized `ytest` value. This component is an `ntree`-by-`nQuantile` matrix where `nQuantile` is the number of elements in `uQuantiles`.
`treeFirstPMFValues`	For each testing set row, this component outputs the pmf value on the minimum (or first) support point in the forest. For binary classification, this corresponds to the probability that the minimum (or first) support point will occur. This component's dimension is `ntest`-by-`ntree`. It is useful for generating calibration curves (stated probabilities in bins vs. their observed frequencies) for binary classification.
`bracketingRate`	For each testing set row, the bracketing rate from Larrick et al. (2012) is computed as `2*p*(1-p)` where `p` is the fraction of trees' means above the `ytest` value. If `ytest` is `NULL`, `NA`s are returned.
`bracketingRateAllPairs`	The average bracketing rate across all testing set rows for each pair of trees. This component is a symmetric `ntree`-by-`ntree` matrix. If `ytest` is `NULL`, `NA`s are returned.

`trimmedEnsembleCDFs`
: For each testing set row, the trimmed ensemble's forecast of `ytest` in the form of a cdf. This component is an `ntest`-by-`nSupport + 1` matrix. `nSupport` is the number of unique `ytrain` values, or support points of the forest.

`trimmedEnsemblePMFs`
: For each testing set row, the trimmed ensemble's pmf. This component is an `ntest`-by-`nSupport` matrix.

`trimmedEnsembleMeans`
: For each testing set row, the trimmed ensemble's mean. This component is an `ntest` vector.

`trimmedEnsembleVars`
: For each testing set row, the trimmed ensemble's variance.

`trimmedEnsemblePITs`
: For each testing set row, the trimmed ensemble's probability integral transform (PIT), the empirical cdf evaluated at the realized `ytest` value. If `ytest` is `NULL`, `NA`s are returned.

`trimmedEnsembleQuantiles`
: For the last testing set row, the trimmed ensemble's quantiles – one for each element in `uQuantiles`.

`trimmedEnsembleComponentScores`
: For the last testing set row, the components of the trimmed ensemble's linear and log quantile scores.If `ytest` is `NULL`, `NA`s are returned.

`trimmedEnsembleScores`
: For each testing set row, the trimmed ensemble's linear and log quantile scores, ranked probability score, and two-moment score. See Jose and Winkler (2009) for a description of the linear and log quantile scores. See Gneiting and Raftery (2007) for a description of the ranked probability score. The two-moment score is the score in Equation 27 of Gneiting and Raftery (2007). If `ytest` is `NULL`, `NA`s are returned.

`untrimmedEnsembleCDFs`
: For each testing set row, the linear opinion pool's, or untrimmed ensemble's, forecast of `ytest` in the form of a cdf.

`untrimmedEnsemblePMFs`
: For each testing set row, the untrimmed ensemble's pmf.

`untrimmedEnsembleMeans`
: For each testing set row, the untrimmed ensemble's mean.

`untrimmedEnsembleVars`
: For each testing set row, the untrimmed ensemble's variance.

`untrimmedEnsemblePITs`
: For each testing set row, the untrimmed ensemble's probability integral transform (PIT), the empirical cdf evaluated at the realized `ytest` value. If `ytest` is `NULL`, `NA`s are returned.

`untrimmedEnsembleQuantiles`
: For the last testing set row, the untrimmed ensemble's quantiles – one for each element in `uQuantiles`.

`untrimmedEnsembleComponentScores`
: For the last testing set row, the components of the untrimmed ensemble's linear and log quantile scores. If `ytest` is `NULL`, `NA`s are returned.

`untrimmedEnsembleScores`
: For each testing set row, the untrimmed ensemble's linear and log quantile scores, ranked probability score, and two-moment score. If `ytest` is `NULL`, `NA`s are returned.

Author(s)

Yael Grushka-Cockayne, Victor Richmond R. Jose, Kenneth C. Lichtendahl Jr., and Huanghui Zeng.

References

Gneiting T, Raftery AE. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association 102 359-378.

Jose VRR, Grushka-Cockayne Y, Lichtendahl KC Jr. (2014). Trimmed opinion pools and the crowd's calibration problem. Management Science 60 463-475.

Jose VRR, Winkler RL (2009). Evaluating quantile assessments. Operations Research 57 1287-1297.

Grushka-Cockayne Y, Jose VRR, Lichtendahl KC Jr. (2014). Ensembles of overfit and overconfident forecasts, working paper.

Larrick RP, Mannes AE, Soll JB (2011). The social psychology of the wisdom of crowds. In J.I. Krueger, ed., Frontiers in Social Psychology: Social Judgment and Decision Making. New York: Psychology Press, 227-242.

See Also

`hitRate`, `cinbag`

Examples

```
# Load the data
set.seed(201) # Can be removed; useful for replication
data <- as.data.frame(mlbench.friedman1(500, sd=1))
summary(data)

# Prepare data for trimming
train <- data[1:400, ]
test <- data[401:500, ]
xtrain <- train[,-11]
ytrain <- train[,11]
xtest <- test[,-11]
ytest <- test[,11]

# Option 1. Run trimTrees with responses in testing set.
set.seed(201) # Can be removed; useful for replication
tt1 <- trimTrees(xtrain, ytrain, xtest, ytest, trim=0.15)

#Some outputs from trimTrees: scores, hit rates, PIT densities.
colMeans(tt1$trimmedEnsembleScores)
colMeans(tt1$untrimmedEnsembleScores)
```

```
mean(hitRate(tt1$treePITs))
hitRate(tt1$trimmedEnsemblePITs)
hitRate(tt1$untrimmedEnsemblePITs)
hist(tt1$trimmedEnsemblePITs, prob=TRUE)
hist(tt1$untrimmedEnsemblePITs, prob=TRUE)

# Option 2. Run trimTrees without responses in testing set.
# In this case, scores, PITs, or hit rates will not be available.
set.seed(201) # Can be removed; useful for replication
tt2 <- trimTrees(xtrain, ytrain, xtest, trim=0.15)

# Some outputs from trimTrees: cdfs for last test value.
plot(tt2$trimmedEnsembleCDFs[100,],type="l",col="red",ylab="cdf",xlab="y")
lines(tt2$untrimmedEnsembleCDFs[100,])
legend(275,0.2,c("trimmed", "untrimmed"),col=c("red","black"),lty = c(1, 1))
title("CDFs of Trimmed and Untrimmed Ensembles")

# Compare the CDF and moment approaches to trimming the trees.
ttCDF <- trimTrees(xtrain, ytrain, xtest, trim=0.15, methodIsCDF=TRUE)
ttMA <- trimTrees(xtrain, ytrain, xtest, trim=0.15, methodIsCDF=FALSE)
plot(ttCDF$trimmedEnsembleCDFs[100,], type="l", col="red", ylab="cdf", xlab="y")
lines(ttMA$trimmedEnsembleCDFs[100,])
legend(275,0.2,c("CDF Approach", "Moment Approach"), col=c("red","black"),lty = c(1, 1))
title("CDFs of Trimmed Ensembles")
```

Index