Package 'Rborist'

September 9, 2019

Title Extensible, Parallelizable Implementation of the Random Forest Algorithm

Version 0.2-2

Date 2019-9-8

Author Mark Seligman

Maintainer Mark Seligman <mseligman@suiji.org>

BugReports https://github.com/suiji/Arborist/issues

SystemRequirements g++ (>= 4.8)

Description Scalable implementation of classification and regression forests, as described by Breiman (2001), <DOI:10.1023/A:1010933404324>.

URL http://www.suiji.org/arborist, https://github.com/suiji/Arborist

License MPL (>= 2) | GPL (>= 2) | file LICENSE

LazyLoad yes

Depends R(>= 3.3)

Imports Rcpp (>= 0.12.2), data.table (>= 1.9.8), digest

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-09-09 08:50:02 UTC

R topics documented:

`Export` *Exportation Format for Rborist Training Output*

Description

Formats training output into a form suitable for illustration of feature contributions.

Usage

```
 ## S3 method for class 'Rborist'
Export(arbOut)
```

Arguments

`arbOut`	an object of type `Rborist` produced by training.

Value

An object of type `Export`.

Author(s)

Mark Seligman at Suiji.

Examples

```
  ## Not run:
    data(iris)
    rb <- Rborist(iris[,-5], iris[,5])
    ffe <- Export(rb)

## End(Not run)
```

ForestFloorExport *Exportation Format for Rborist Training Output*

Description

Formats training output into a form suitable for `ForestFloor` feature-contribution package.

Usage

```
 ## S3 method for class 'Rborist'
ForestFloorExport(arbOut)
```

Arguments

`arbOut`	an object of type `Rborist` produced by training.

Value

An object of type `ForestFloorExport`, as specified by the interface for the `ForestFloor` package.

Author(s)

Mark Seligman at Suiji.

Examples

```
  ## Not run:
    data(iris)
    rb <- Rborist(iris[,-5], iris[,5])
    ffe <- ForestFloorExport(rb)

    library(ForestFloor)
    ForestFloor(ffe)

## End(Not run)
```

predict.Rborist *predict method for Rborst*

Description

Prediction and test using Rborist.

Usage

```
## S3 method for class 'Rborist'
predict(object, newdata, yTest=NULL, quantVec=NULL,
quantiles = !is.null(quantVec), ctgCensus = "votes", oob = FALSE,
nThread = 0, verbose = FALSE, ...)
```

Arguments

object	an object of class Rborist, created from a previous invocation of the command Rborist to train.
newdata	a design matrix containing new data, with the same signature of predictors as in the training command.
yTest	if specfied, a response vector against which to test the new predictions.
quantVec	a vector of quantiles to predict.
quantiles	whether to predict quantiles.
ctgCensus	whether/how to summarize per-category predictions. "votes" specifies the number of trees predicting a given class. "prob" specifies a normalized, probabilistic summary.
oob	whether prediction is restricte to out-of-bag samples.
nThread	suggests ans OpenMP-style thread count. Zero denotes default processor setting.
verbose	whether to output progress of prediction.
...	not currently used.

Value

a list containing either of the two prediction containers:

PredictReg	a list of prediction results for regression:
	yPred a vector containing the predicted response.
	qPred a matrix containing the prediction quantiles, if requested.
PredictCtg	a list of validation results for classification:
	yPred a vector containing the predicted response.
	census a matrix of predictions, by category.
	prob a matrix of prediction probabilities by category, if requested.

Author(s)

Mark Seligman at Suiji.

See Also

Rborist

Examples

```
## Not run:
  # Regression example:
  nRow <- 5000
  x <- data.frame(replicate(6, rnorm(nRow)))
  y <- with(x, X1^2 + sin(X2) + X3 * X4) # courtesy of S. Welling.
  rb <- Rborist(x,y)


  # Performs separate prediction on new data:
  xx <- data.frame(replace(6, rnorm(nRow)))
  pred <- predict(rb, xx)
  yPred <- pred$yPred


  # Performs separate prediction, using original response as test
  # vector:
  pred <- predict(rb, xx, y)
  mse <- pred$mse
  rsq <- pred$rsq


  # Performs separate prediction with (default) quantiles:
  pred <- predict(rb, xx, quantiles="TRUE")
  qPred <- pred$qPred


  # Performs separate prediction with deciles:
  pred <- predict(rb, xx, quantVec = seq(0.1, 1.0, by = 0.10))
  qPred <- pred$qPred


  # Classification examples:
  data(iris)
  rb <- Rborist(iris[-5], iris[5])


  # Generic prediction using training set.
  # Census as (default) votes:
  pred <- predict(rb, iris[-5])
  yPred <- pred$yPred
  census <- pred$census


  # As above, but validation census to report class probabilities:
  pred <- predict(rb, iris[-5], ctgCensus="prob")
  prob <- pred$prob


  # As above, but with training reponse as test vector:
  pred <- predict(rb, iris[-5], iris[5], ctgCensus = "prob")
  prob <- pred$prob
```

```
  conf <- pred$confusion
  misPred <- pred$misPred

## End(Not run)
```

`PreFormat` *Preformatting for Training with Warm Starts*

Description

Presorts and formats training input into a form suitable for subsequent training by `Rborist` command. Saves unnecessary recomputation of this form when iteratively retraining.

Usage

```
## Default S3 method:
PreFormat(x, verbose)
```

Arguments

`x`	the design matrix expressed as either a `data.frame` object with numeric and/or `factor` columns or as a numeric matrix.
`verbose`	indicates whether to output progress of preformatting.

Value

`PreFormat`	a list consisting of three objects:
	`predFrame` a list of presorted and formatted predictor information:
	`blockFac` a matrix of zero-based factor predictor values.
	`blockFacSparse` a sparse block of factor predictor values.
	`blockNum` a matrix of dense numeric predictor values.
	`blockNumSparse` a sparse block of numeric predictor values.
	`nPredFac` the number of factor predictors.
	`nRow` the number of training rows.
	`nPredFac` the number of factor-valued predictors.
	`facCard` a vector of the factor cardinalities.
	`nPredNum` the number of numerical predictors.
	`summaryRLE` a run-length encoded version of the training observations. `obsHash` a hashed version of the training observations, used to ensure that subsequent invocations of Rborist employ the same observations used to train.

Author(s)

Mark Seligman at Suiji.

Examples

```
## Not run:
  data(iris)
  pt <- PreFormat(iris[,-5])

  ppTry <- seq(0.2, 0.5, by= 0.3/10)
  nIter <- length(ppTry)
  rsq <- numeric(nIter)
  for (i in 1:nIter) {
    rb <- Rborist(pt, iris[,5], predProb=ppTry[i])
    rsq[i] = rb$validiation$rsq
  }

## End(Not run)
```

Rborist *Rapid Decision Tree Construction and Evaluation*

Description

Accelerated implementation of the Random Forest (trademarked name) algorithm. Tuned for multicore and GPU hardware. Bindable with most numerical front-end languages in addtion to R. Invocation is similar to that provided by "randomForest" package.

Usage

```
## Default S3 method:
 Rborist(x,
                y,
                autoCompress = 0.25,
                ctgCensus = "votes",
                classWeight = NULL,
                maxLeaf = 0,
                minInfo = 0.01,
                minNode = ifelse(is.factor(y), 2, 3),
                nLevel = 0,
                nSamp = 0,
                nThread = 0,
                nTree = 500,
                noValidate = FALSE,
                predFixed = 0,
                predProb = 0.0,
                predWeight = NULL,
                quantVec = NULL,
                quantiles = !is.null(quantVec),
                regMono = NULL,
                rowWeight = NULL,
```

```
splitQuant = NULL,
thinLeaves = ifelse(is.factor(y), TRUE, FALSE),
treeBlock = 1,
verbose = FALSE,
withRepl = TRUE,
...)
```

Arguments

x	the design matrix expressed as a PreFormat object, as a data.frame object with numeric and/or factor columns or as a numeric matrix.
y	the response (outcome) vector, either numerical or categorical. Row count must conform with x.
autoCompress	plurality above which to compress predictor values.
ctgCensus	report categorical validation by vote or by probability.
classWeight	proportional weighting of classification categories.
maxLeaf	maximum number of leaves in a tree. Zero denotes no limit.
minInfo	information ratio with parent below which node does not split.
minNode	minimum number of distinct row references to split a node.
nLevel	maximum number of tree levels to train. Zero denotes no limit.
nSamp	number of rows to sample, per tree.
nThread	suggests an OpenMP-style thread count. Zero denotes the default processor setting.
nTree	the number of trees to train.
noValidate	whether to train without validation.
predFixed	number of trial predictors for a split (mtry).
predProb	probability of selecting individual predictor as trial splitter.
predWeight	relative weighting of individual predictors as trial splitters.
quantVec	quantile levels to validate.
quantiles	whether to report quantiles at validation.
regMono	signed probability constraint for monotonic regression.
rowWeight	row weighting for initial sampling of tree.
splitQuant	(sub)quantile at which to place cut point for numerical splits.
thinLeaves	bypasses creation of export and quantile state in order to reduce memory footprint.
treeBlock	maximum number of trees to train during a single level (e.g., coprocessor computing).
verbose	indicates whether to output progress of training.
withRepl	whether row sampling is by replacement.
...	not currently used.

Value

an object of class `Rborist`, a list containing the following items:

`forest` a list containing

`forestNode` a vector of packed structures expressing splitting predictors, splitting values, successor node deltas and leaf indices.

`height` a vector of accumulated tree heights within `forestNode`.

`facSplit` a vector of splitting factor values.

`facHeight` a vector of accumulated tree heights positions within the splitting factor values.

a list containing either of: `LeafReg` a list consisting of regression leaf data:

`node` a packed structure expressing leaf scores and node counts.

`nodeHeight` a vector of accumulated tree heights within `node`.

`bagHeight` a vector of accumulated bag counts, per tree.

`bagSample` a vector of packed data structures, one per unique row sample, containing the row index and number of times sampled.

`yTrain` the training response.

or `LeafCtg` a list consisting of classification leaf data:

`node` a packed structure expressing leaf scores and node counts.

`nodeHeight` a vector of accumulated tree heights within `node`.

`bagHeight` a vector of accumulated bag counts, per tree.

`bagSample` a vector of packed data structures, one per unique row sample, containing the row index and number of times sampled. `weight` a vector of per-category probabilities, one set for each sampled row.

`levels` a vector of strings containing the training response levels.

`bag` a list consisting of bagged row information:

`raw` a packed bit matrix indicating whether a given row, tree pair is bagged.

`nRow` the number of rows employed in training.

`nTree` the number of trained trees.

`rowBytes` the row stride, in bytes.

`training` a list containing information gleaned during training:

`call` a string containing the original invocation.

`info` the information contribution of each predictor.

`version` the version of the Rborist package.

`diag` strings containing unspecified diagnostic notes and observations.

`validation` a list containing the results of validation, if requested:

`ValidReg` a list of validation results for regression:

`yPred` vector containing the predicted response.

`mae` the mean absolute error of prediction.

`mse` the mean-square error of prediction.

rsq the r-squared statistic.
qPred matrix containing the prediction quantiles, if requested.
ValidCtg list of validation results for classification:
yPred vector containing the predicted response.
misprediction vector containing the classwise misprediction rates.
confusion the confusion matrix.
census matrix of predictions, by category.
oobError the out-of-bag error.
prob matrix of prediction probabilities by category, if requested.

Author(s)

Mark Seligman at Suiji.

Examples

```
## Not run:
  # Regression example:
  nRow <- 5000
  x <- data.frame(replicate(6, rnorm(nRow)))
  y <- with(x, X1^2 + sin(X2) + X3 * X4) # courtesy of S. Welling.

  # Classification example:
  data(iris)

  # Generic invocation:
  rb <- Rborist(x, y)


  # Causes 300 trees to be trained:
  rb <- Rborist(x, y, nTree = 300)


  # Causes rows to be sampled without replacement:
  rb <- Rborist(x, y, withRepl=FALSE)


  # Causes validation census to report class probabilities:
  rb <- Rborist(iris[-5], iris[5], ctgCensus="prob")


  # Applies table-weighting to classification categories:
  rb <- Rborist(iris[-5], iris[5], classWeight = "balance")


  # Weights first category twice as heavily as remaining two:
  rb <- Rborist(iris[-5], iris[5], classWeight = c(2.0, 1.0, 1.0))


  # Does not split nodes when doing so yields less than a 2% gain in
  # information over the parent node:
```

```
rb <- Rborist(x, y, minInfo=0.02)


# Does not split nodes representing fewer than 10 unique samples:
rb <- Rborist(x, y, minNode=10)


# Trains a maximum of 20 levels:
rb <- Rborist(x, y, nLevel = 20)


# Trains, but does not perform subsequent validation:
rb <- Rborist(x, y, noValidate=TRUE)


# Chooses 500 rows (with replacement) to root each tree.
rb <- Rborist(x, y, nSamp=500)


# Chooses 2 predictors as splitting candidates at each node (or
# fewer, when choices exhausted):
rb <- Rborist(x, y, predFixed = 2)


# Causes each predictor to be selected as a splitting candidate with
# distribution Bernoulli(0.3):
rb <- Rborist(x, y, predProb = 0.3)


# Causes first three predictors to be selected as splitting candidates
# twice as often as the other two:
rb <- Rborist(x, y, predWeight=c(2.0, 2.0, 2.0, 1.0, 1.0))


# Causes (default) quantiles to be computed at validation:
rb <- Rborist(x, y, quantiles=TRUE)
qPred <- rb$validation$qPred


# Causes specfied quantiles (deciles) to be computed at validation:
rb <- Rborist(x, y, quantVec = seq(0.1, 1.0, by = 0.10))
qPred <- rb$validation$qPred


# Constrains modelled response to be increasing with respect to X1
# and decreasing with respect to X5.
rb <- Rborist(x, y, regMono=c(1.0, 0, 0, 0, -1.0, 0))


# Causes rows to be sampled with random weighting:
rb <- Rborist(x, y, rowWeight=runif(nRow))
```

```
  # Suppresses creation of detailed leaf information needed for
  # quantile prediction and external tools.
  rb <- Rborist(x, y, thinLeaves = TRUE)


  # Sets splitting position for predictor 0 to far left and predictor
  # 1 to far right, others to default (median) position.

  spq <- rep(0.5, ncol(x))
  spq[0] <- 0.0
  spq[1] <- 1.0
  rb <- Rborist(x, y, splitQuant = spq)

## End(Not run)
```

`RboristNews` *NEWS Displayer for Rborist*

Description

Displays NEWS associated with Rborist releases.

Usage

```
RboristNews()
```

Value

None.

`Streamline` *Reducing Memory Footprint of Trained Decision Forest*

Description

Clears fields deemed no longer useful.

Usage

```
## Default S3 method:
Streamline(rs)
```

Arguments

`rs`	Trained forest object.

Value

an object of class Rborist with certain fields cleared.

Author(s)

Mark Seligman at Suiji.

Examples

```
## Not run:
    ## Trains.
    rs <- Rborist(x, y)
    ...
    ## Replaces trained object with streamlined copy.
    rs <- Streamline(rs)

## End(Not run)
```

Validate *Separate Validation of Trained Decision Forest*

Description

Permits trained decision forest to be validated separately from training.

Usage

```
## Default S3 method:
Validate(preFormat, train, y, ctgCensus = "votes",
quantVec = NULL, quantiles = !is.null(quantVec), nThread = 0, verbose = FALSE)
```

Arguments

preFormat	internal representation of the design matrix, of class PreFormat
train	an object of class Rborist obtained from previous training.
y	the response (outcome) vector, either numerical or categorical. Row count must conform with x.
ctgCensus	report categorical validation by vote or by probability.
quantVec	quantile levels to validate.
quantiles	whether to report quantiles at validation.
nThread	suggests an OpenMP-style thread count. Zero denotes the default processor setting.
verbose	indicates whether to output progress of validation.

Value

an object of class `validation`:

`validation` list containing either a:
`ValidReg` list of validation results for regression:
`yPred` vector containing the predicted response.
`mae` the mean absolute error of prediction.
`mse` the mean-square error of prediction.
`rsq` the r-squared statistic.
`qPred` matrix containing the prediction quantiles, if requested.
or a: `ValidCtg` list of validation results for classification:
`yPred` vector containing the predicted response.
`misprediction` vector containing the classwise misprediction rates.
`confusion` the confusion matrix.
`census` matrix of predictions, by category.
`oobError` the out-of-bag error.
`prob` matrix of prediction probabilities by category, if requested.

Author(s)

Mark Seligman at Suiji.

Examples

```
## Not run:
    ## Trains without validation.
    rb <- Rborist(x, y, noValidate=TRUE)
    ...
    ## Delayed validation using a PreFormat object.
    pf <- PreFormat(x)
    v <- Validate(pf, rb, y)

## End(Not run)
```

Index