Package ‘extraTrees’

February 19, 2015

Version 1.0.5

Date 2014-12-27

Title Extremely Randomized Trees (ExtraTrees) Method for Classification and Regression

Author Jaak Simm, Ildefons Magrans de Abril

Maintainer Jaak Simm <jaak.simm@gmail.com>

Description Classification and regression based on an ensemble of decision trees. The package also provides extensions of ExtraTrees to multi-task learning and quantile regression. Uses Java implementation of the method.

Depends R (>= 2.7.0), rJava (>= 0.5-0)

Suggests testthat, Matrix

SystemRequirements Java (>= 1.6)

NeedsCompilation no

License Apache License 2.0

URL http://github.com/jaak-s/extraTrees

Repository CRAN

Date/Publication 2014-12-27 23:41:04

R topics documented:

extraTrees *Function for training ExtraTree classifier or regression.*

Description

This function executes ExtraTree building method (implemented in Java).

Usage

```
  ## Default S3 method:
extraTrees(x, y,
             ntree=500,
             mtry = if (!is.null(y) && !is.factor(y))
                    max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             numRandomCuts = 1,
             evenCuts = FALSE,
             numThreads = 1,
             quantile = F,
             weights = NULL,
             subsetSizes = NULL,
             subsetGroups = NULL,
             tasks = NULL,
             probOfTaskCuts = mtry / ncol(x),
             numRandomTaskCuts = 1,
             na.action = "stop",
             ...)
```

Arguments

`x` a numberic input data matrix, each row is an input.

`y` a vector of output values: if vector of numbers then regression, if vector of factors then classification.

`ntree` the number of trees (default 500).

`mtry` the number of features tried at each node (default is ncol(x)/3 for regression and sqrt(ncol(x)) for classification).

`nodesize` the size of leaves of the tree (default is 5 for regression and 1 for classification)

`numRandomCuts` the number of random cuts for each (randomly chosen) feature (default 1, which corresponds to the official ExtraTrees method). The higher the number of cuts the higher the chance of a good cut.

`evenCuts` if FALSE then cutting thresholds are uniformly sampled (default). If TRUE then the range is split into even intervals (the number of intervals is numRandomCuts) and a cut is uniformly sampled from each interval.

`numThreads` the number of CPU threads to use (default is 1).

`quantile` if TRUE then quantile regression is performed (default is FALSE), only for regression data. Then use predict(et, newdata, quantile=k) to make predictions for k quantile.

`weights` a vector of sample weights, one positive real value for each sample. NULL means standard learning, i.e. equal weights.

`subsetSizes` subset size (one integer) or subset sizes (vector of integers, requires subsetGroups), if supplied every tree is built from a random subset of size subsetSizes. NULL means no subsetting, i.e. all samples are used.

`subsetGroups` list specifying subset group for each sample: from samples in group g, each tree will randomly select subsetSizes[g] samples.

`tasks` vector of tasks, integers from 1 and up. NULL if no multi-task learning

`probOfTaskCuts` probability of performing task cut at a node (default mtry / ncol(x)). Used only if tasks is specified.

`numRandomTaskCuts`
number of times task cut is performed at a node (default 1). Used only if tasks is specified.

`na.action` specifies how to handle NA in x: "stop" (default) will give error is any NA present, "zero" will set all NA to zero and "fuse" will build trees by skipping samples when the chosen feature is NA for them.

`...` not used currently.

Details

For classification ExtraTrees at each node chooses the cut based on minimizing the Gini impurity index and for regression the variance.

For more details see the package vignette, i.e. vignette("extraTrees").

If Java runs out of memory: `java.lang.OutOfMemoryError: Java heap space`, then (assuming you have free memory) you can increase the heap size by: `options( java.parameters = "-Xmx2g" )` before calling `library( "extraTrees" )`, where 2g defines 2GB of heap size. Change it as necessary.

Value

The trained model from input x and output values y, stored in ExtraTree object.

Author(s)

Jaak Simm

See Also

`predict.extraTrees` for predicting and `prepareForSave` for saving ExtraTrees models to disk.

Examples

```
## Regression with ExtraTrees:
n <- 1000  ## number of samples
p <- 5     ## number of dimensions
x <- matrix(runif(n*p), n, p)
y <- (x[,1]>0.5) + 0.8*(x[,2]>0.6) + 0.5*(x[,3]>0.4) +
     0.1*runif(nrow(x))
et <- extraTrees(x, y, nodesize=3, mtry=p, numRandomCuts=2)
yhat <- predict(et, x)

#######################################
## Multi-task regression with ExtraTrees:
n <- 1000  ## number of samples
p <- 5     ## number of dimensions
x <- matrix(runif(n*p), n, p)
task <- sample(1:10, size=n, replace=TRUE)
## y depends on the task:
y <- 0.5*(x[,1]>0.5) + 0.6*(x[,2]>0.6) + 0.8*(x[cbind(1:n,(task %% 2) + 3)]>0.4)
et <- extraTrees(x, y, nodesize=3, mtry=p-1, numRandomCuts=2, tasks=task)
yhat <- predict(et, x, newtasks=task)

#######################################
## Classification with ExtraTrees (with test data)
make.data <- function(n) {
  p <- 4
  f <- function(x) (x[,1]>0.5) + (x[,2]>0.6) + (x[,3]>0.4)
  x <- matrix(runif(n*p), n, p)
  y <- as.factor(f(x))
  return(list(x=x, y=y))
}
train <- make.data(800)
test  <- make.data(500)
et    <- extraTrees(train$x, train$y)
yhat  <- predict(et, test$x)
## accuracy
mean(test$y == yhat)
## class probabilities
yprob = predict(et, test$x, probability=TRUE)
head(yprob)

#######################################
## Quantile regression with ExtraTrees (with test data)
make.qdata <- function(n) {
  p <- 4
  f <- function(x) (x[,1]>0.5) + 0.8*(x[,2]>0.6) + 0.5*(x[,3]>0.4)
  x <- matrix(runif(n*p), n, p)
  y <- as.numeric(f(x))
  return(list(x=x, y=y))
}
train <- make.qdata(400)
test  <- make.qdata(200)
```

```
## learning extra trees:
et <- extraTrees(train$x, train$y, quantile=TRUE)
## estimate median (0.5 quantile)
yhat0.5 <- predict(et, test$x, quantile = 0.5)
## estimate 0.8 quantile (80%)
yhat0.8 <- predict(et, test$x, quantile = 0.8)


#######################################
## Weighted regression with ExtraTrees
make.wdata <- function(n) {
  p <- 4
  f <- function(x) (x[,1]>0.5) + 0.8*(x[,2]>0.6) + 0.5*(x[,3]>0.4)
  x <- matrix(runif(n*p), n, p)
  y <- as.numeric(f(x))
  return(list(x=x, y=y))
}
train <- make.wdata(400)
test  <- make.wdata(200)

## first half of the samples have weight 1, rest 0.3
weights <- rep(c(1, 0.3), each = nrow(train$x) / 2)
et <- extraTrees(train$x, train$y, weights = weights, numRandomCuts = 2)
## estimates of the weighted model
yhat <- predict(et, test$x)
```

`predict.extraTrees` *Function for making predictions from trained ExtraTree object.*

Description

This function makes predictions for regression/classification using the given trained ExtraTree object and provided input matrix (newdata).

Usage

```
  ## S3 method for class 'extraTrees'
predict(object, newdata, quantile=NULL, allValues=F, probability=F, newtasks=NULL, ...)
```

Arguments

`object` extraTree (S3) object, created by extraTrees().

`newdata` a new numberic input data matrix, for each row a prediction is made.

`quantile` the quantile value between 0.0 and 1.0 for quantile regression, or NULL (default) for standard predictions.

`allValues` whether or not to return outputs of all trees (default FALSE).

`probability` whether to return a matrix of class (factor) probabilities, default FALSE. Can only be used in the case of classification. Calculated as the proportion of trees voting for particular class.

`newtasks` list of tasks, for each input in newdata (default NULL). Must be NULL if no multi-task learning was used at training.

`...` not used currently.

Value

The vector of predictions from the ExtraTree et. The length of the vector is equal to the the number of rows in newdata.

Author(s)

Jaak Simm

Examples

```
## Regression with ExtraTrees:
n <- 1000  ## number of samples
p <- 5     ## number of dimensions
x <- matrix(runif(n*p), n, p)
y <- (x[,1]>0.5) + 0.8*(x[,2]>0.6) + 0.5*(x[,3]>0.4) + 0.1*runif(nrow(x))
et <- extraTrees(x, y, nodesize=3, mtry=p, numRandomCuts=2)
yhat <- predict(et, x)
```

prepareForSave *Prepares ExtraTrees object for save() function*

Description

This function prepares ExtraTrees for saving by serializing the trees in Java VM. It is equivalent to calling `.jcache(et$jobject)`. Afterwards the object can be saved by `save` (or automatic R session saving) and will be fully recovered after `load`.

Note: the object can still be used as usual after `prepareForSave`.

Usage

```
prepareForSave(object)
```

Arguments

`object` extraTrees (S3) object, created by extraTrees().

Value

Nothing is returned.

Author(s)

Jaak Simm

Examples

```
et <- extraTrees(iris[,1:4], iris$Species)
prepareForSave(et)
## saving to a file
save(et, file="temp.Rdata")

## testing: remove et and load it back from file
rm(list = "et")
load("temp.Rdata")
predict(et, iris[,1:4])
```

selectTrees *Makes a sub-ExtraTrees object by keeping only selected trees.*

Description

This function creates a sub-ExtraTrees object by keeping only selected trees specified by selection.

Usage

```
selectTrees(object, selection)
```

Arguments

object	extraTrees (S3) object, created by extraTrees().
selection	a list of logicals (T/F) of length object$ntree.

Value

A new ExtraTrees (S3) object based on the existing object by keeping only the trees present in the selection.

Author(s)

Jaak Simm

Examples

```
## Regression with ExtraTrees:
n <- 1000  ## number of samples
p <- 5     ## number of dimensions
x <- matrix(runif(n*p), n, p)
y <- (x[,1]>0.5) + 0.8*(x[,2]>0.6) + 0.5*(x[,3]>0.4) + 0.1*runif(nrow(x))
et <- extraTrees(x, y, nodesize=3, mtry=p, numRandomCuts=2, ntree=500)
## random selection of trees:
trees <- sample(c(FALSE, TRUE), replace=TRUE, et$ntree)
et2   <- selectTrees(et, selection=trees)
```

setJavaMemory *Utility function for setting Java memory.*

Description

Function for setting JVM memory, specified in MB. If you get java.lang.OutOfMemoryError you can use this function to increase the memory available to ExtraTrees.

Usage

```
setJavaMemory( memoryInMB )
```

Arguments

memoryInMB	Integer specifying the amount of memory (MB)

Author(s)

Jaak Simm

Examples

```
## use 2G memory
setJavaMemory(2000)
```

toJavaCSMatrix *Utility function for converting an R SparseMatrix (package Matrix) to Java (column) sparse matrix.*

Description

Internal function used for converting an R SparseMatrix (package Matrix) to a CSparseMatrix object in Java. CSparseMatrix class is a custom Java class used for storing sparse matrices by the implementation of ExtraTrees in Java.

Usage

```
toJavaCSMatrix( m )
```

Arguments

m	matrix of numeric values.

Value

reference to Java matrix with the same contents as the input R matrix.

Author(s)

Jaak Simm

`toJavaMatrix` *Utility function for converting an R matrix (numeric matrix) to Java matrix.*

Description

Internal function used for converting an R matrix to a Matrix object in Java. Matrix class is a custom Java class used for storing matrices by the implementation of ExtraTrees in Java.

Usage

```
toJavaMatrix( m )
```

Arguments

`m`	matrix of numeric values.

Value

reference to Java matrix with the same contents as the input R matrix.

Author(s)

Jaak Simm

`toJavaMatrix2D` *Utility function for converting an R matrix (standard matrix or SparseMatrix) to appropriate Java matrix object.*

Description

Internal function used for converting an R matrix to an appropriate object in Java. It uses toJavaMatrix() and toJavaCSMatrix() underneath and returns a reference to general matrix representation in Java of type Array2D (interface).

Usage

```
toJavaMatrix2D( m )
```

Arguments

`m`	matrix of numeric values.

Value

reference to Java matrix (dense or sparse) with the same contents as the input R matrix.

Author(s)

Jaak Simm

---

toRMatrix *Utility function for converting Java matrix to R matrix (matrix of doubles).*

Description

Internal function used for converting a Matrix object from Java to an R matrix. Matrix class is a custom Java class used for storing matrices by the implementation of ExtraTrees in Java.

Usage

```
toRMatrix( javam )
```

Arguments

javam	Java matrix (Matrix class).

Value

R (double) matrix with the same contents as the input.

Author(s)

Jaak Simm

Index