Package 'HPdclassifier'

January 20, 2015

Type Package

Title Distributed classifiers for Big Data

Version 1.0.0

Date 2015-01-16

Author HP Vertica Analytics Team

Maintainer HP Vertica Analytics Team <distributedRTeam@external.groups.hp.com>

Depends R (>= 3.0.0), distributedR, randomForest (>= 4.6-10)

Description

Distributed algorithms for learning classifiers. Written using HP Vertica Distributed R package.

License GPL (>= 2) | file LICENSE

R topics documented:

`HPdclassifier-package`

Distributed algorithms for classifiers

Description

HPdclassifier provides several distributed algorithms for classifiers. It is written based on the infrastructure created in HP-Labs for distributed computing in R.

Details

Package:	HPdclassifier
Type:	Package
Version:	1.0.0
Date:	2015-01-16

Main Functions:

- hpdrandomForest: It is a distributed function for randomForest
- predictHPdRF: distributed predict method for applying a random forest objects on a darray or a dframe

Author(s)

HP Vertica Analytics Team <distributedRTeam@external.groups.hp.com>

References

1. Using R for Iterative and Incremental Processing. Shivaram Venkataraman, Indrajit Roy, Alvin AuYoung, Rob Schreiber. HotCloud 2012, Boston, USA.

`hpdrandomForest` *Distributed randomForest*

Description

hpdrandomForest function runs `randomForest` function of randomForest package in a distributed fashion.

Description

`hpdrandomForest` calls several instances of `randomForest` distributed across a cluster system. Therefore, the master distributes the input data among all R-executors of the distributedR environment, and trees on different sub-sections of the forest are created simultaneously. At the end, all these trees are combined to result a single forest.

The interface of `hpdrandomForest` is similar to `randomForest`. Indeed it adds two arguments `nExecutor` and `trace`, and removes several other arguments `do.trace`, `corr.bias`, `keep.inbag`, and `oob.prox`. Its returned result is also completely compatible to the result of `randomForest`.

Usage

```
## S3 method for class 'formula'
hpdrandomForest(formula, data=NULL, ..., ntree=500,
         na.action=na.fail, nExecutor, trace=FALSE,
         completeModel=FALSE)
## Default S3 method:
hpdrandomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
         mtry=if (!is.null(y) && !is.factor(y) && !is.dframe(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) &&
         !is.dframe(y)) 5 else 1,
         maxnodes=NULL, importance=FALSE, localImp=FALSE, nPerm=1,
         proximity=FALSE,norm.votes=TRUE, keep.forest=TRUE,
         nExecutor, trace=FALSE, completeModel=FALSE, ...)
```

Arguments

data	a data frame or dframe which contains samples.
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)
formula	a formula describing the model to be fitted.
x	when a data frame or a matrix of predictors assigned to x, its size should not be bigger than 2GB. For bigger datasets, darray should be used. darray does not support categorical data. Therefore, dframe and the first interface should be used for classification problems of large datasets with categorical data.
y	a response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, hpdrandomForest will run in unsupervised mode. When x is a darray), y should be also a darray with a single column.
xtest	a data frame or matrix (like x) containing predictors for the test set. When x is a darray, it should be of the same type.
ytest	response for the test set. Its type should be consistent with y. Moreover, it should have a single column.
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	a logical value which indicates if the proximity measure among the rows should be calculated. It is FALSE by default because it is very memory inefficient. Moreover, it is 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.
keep.forest	If set to FALSE, the forest will not be retained in the output object.
nExecutor	a positive integer number indicating the number of tasks for running the function. To have optimal performance, it is recommended to have this number smaller than the number of R-executors in the distributedR environment. It cannot be bigger than ntree.
trace	when this argument is true, intermediate steps of the progress are displayed.
completeModel	when it is FALSE (default), the output values that preserve information per sample are discarded. They are 'y', 'oob.times', 'votes', 'predicted', and 'test'. This feature is intended to keep the size of the output model small. \item...optional parameters to be passed to the low level function.

Value

An object of class randomForest. The result is similar to the result of the combine function in randomForest package and will contain the following components.

call	the original call to hpdrandomForest
type	one of regression, classification, or unsupervised.
predicted	(only when completeModel=TRUE) 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 hpdrandomForest 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, and only when completeModel=TRUE) 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	(only when completeModel=TRUE) number of times cases are 'out-of-bag' (and thus used in computing OOB error estimate)

y	(only when completeModel=TRUE) the response vector if it is made available in the input.
proximity	(only when completeModel=TRUE) a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes) when proximity=TRUE.
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	(only when completeModel=TRUE) 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.
terms	it contains a formula identifying response and predictors (for classification and regression types).

Note

When ntree is not big enough in comparison to nExecutor, some of the returned predicted values may become NULL. It is the same for values of 'votes' matrix when they are normalized (norm.votes=TRUE). Returned values for err.rate, votes, and oob.times are valid only for classification type.

Three scenarios can be imagined for the type of input data. When ordinary R types are used (matrix or data.frame) the behavior is similar to the randomForest function; however, the total size of the input data cannot be bigger than 2GB. In fact, for bigger data size distributed data types; i.e., darray or dframe, should be used. When x is of type darray, in the case of existence y must be of type darray as well. Regarding the fact that darray does not support categorical data, this data type cannot be used for classification mode. When x is of type dframe, no value can be assigned to y; indeed for this data type, the formula interface should be used for classification and the default interface for unsupervised mode.

Author(s)

HP Vertica Analytics Team

References

Breiman, L. (2001), *Random Forests*, Machine Learning 45(1),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.

Random Forests V4.6-10, http://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

Examples

```
 ## Not run:

library(HPdclassifier)
distributedR_start()
drs <- distributedR_status()
nparts <- sum(drs$Ins)
```

```
## Classification:
##data(iris)
iris.rf <- hpdrandomForest(Species ~ ., data=iris, importance=TRUE,
                       nExecutor=nparts)
print(iris.rf)

## The 'unsupervised' case:
iris.urf <- hpdrandomForest(iris[, -5], nExecutor=nparts,
                            proximity=TRUE, completeModel=TRUE)
MDSplot(iris.urf, iris$Species)

## stratified sampling: draw 20, 30, and 20 of the species to grow each tree.
(iris.rf2 <- hpdrandomForest(iris[1:4], iris$Species,
                         sampsize=c(20, 30, 20), nExecutor=nparts))

## Regression:
## data(airquality)
ozone.rf <- hpdrandomForest(Ozone ~ ., data=airquality, mtry=3,
                    importance=TRUE, na.action=na.omit,
                    nExecutor=nparts, completeModel=TRUE)
print(ozone.rf)
## Show "importance" of variables: higher value mean more important:
round(importance(ozone.rf), 2)

## "x" can be a matrix instead of a data frame:
x <- matrix(runif(5e2), 100)
y <- gl(2, 50)
(myrf <- hpdrandomForest(x, y, nExecutor=nparts))
(predict(myrf, x))

## "complicated" formula:
(swiss.rf <- hpdrandomForest(sqrt(Fertility)~. - Catholic + I(Catholic<50),
                         data=swiss, nExecutor=nparts))
(predict(swiss.rf, swiss))
## Test use of 32-level factor as a predictor:
x <- data.frame(x1=gl(32, 5), x2=runif(160), y=rnorm(160))
(rf1 <- hpdrandomForest(x[-3], x[[3]], ntree=10, nExecutor=nparts))

## Grow no more than 4 nodes per tree:
(treesize(hpdrandomForest(Species ~ ., data=iris, maxnodes=4, ntree=30,
                                nExecutor=nparts)))

distributedR_shutdown()

## End(Not run)
```

`predictHPdRF` *distributed predict method for applying a random forest objects on a darray or a dframe*

Description

Prediction of distributed test data using random forest.

Usage

```
predictHPdRF(object, newdata, trace=FALSE)
```

Arguments

object	an object of class randomForest, as that created by the function randomForest or hpdrandomForest.
newdata	a darray or a dframe containing new data. darray is highly recommended when there is no categorial data
trace	when this argument is true, intermediate steps of the progress are displayed.

Value

It returns a darray or a dframe of predicted classes. When the newdata is of type darray, the type of returned value will be also darray unless the output is categorical data. When the output is a dframe when the newdata is of type dframe.

Author(s)

HP Vertica Analytics Team

References

Breiman, L. (2001), *Random Forests*, Machine Learning 45(1), 5-32.

See Also

hpdrandomForest

Examples

```
 ## Not run:
# example for darray
library(HPdclassifier)
distributedR_start()
drs <- distributedR_status()
nparts <- sum(drs$Ins)

nSamples <- 100
nAttributes <- 5
nPartitions <- 2

dax <- darray(c(nSamples,nAttributes),
              c(ceiling(nSamples/nPartitions),nAttributes))
day <- darray(c(nSamples,1), c(ceiling(nSamples/nPartitions),1))

foreach(i,1:npartitions(dax),function(x=splits(dax,i),
                                      y=splits(day,i),id=i){
    x <- matrix(runif(nrow(x)*ncol(x)), nrow(x),ncol(x))
    y <- matrix(runif(nrow(y)), nrow(y), 1)
    update(x)
    update(y)
})

(myrf1 <- hpdrandomForest(dax, day, nExecutor=nparts))
```

```
dp <- predictHPdRF(myrf1, dax)

# example for dframe
nSamples <- nrow(iris)
nAttributes <- ncol(iris)
nPartitions <- 4

df <- dframe(c(nSamples,nAttributes),
             c(ceiling(nSamples/nPartitions),nAttributes))
end = cumsum(partitionsize(df)[,1])
start = c(0,end[-length(end)])

foreach(i, 1:npartitions(df), function(xi=splits(df,i),
  start = start[i]+1, end = end[i], iris=iris){
     xi <- iris[start:end,]
     update(xi)
})
# the following line will be redundant in the next release
colnames(df) <- colnames(iris)

(myrf2 <- hpdrandomForest(Species ~ ., df, nExecutor=nPartitions))
fp2 <- predictHPdRF(myrf2, df)


## End(Not run)
```

Index