Package 'h2o'

July 31, 2013

R topics documented:

`h2o-package` *H2O R Interface*

Description

This is a package for running H2O via its REST API from within R.

Details

Package:	h2o
Type:	Package
Version:	99.99.99.99999
Date:	2013-07-16
License:	Apache-2
Depends:	R (>= 2.13.0), RCurl, rjson

This package allows the user to run basic H2O commands using R commands. In order to use it, you must first have H2O running (See How to Start H2O). Load the library and create a `H2OClient` object with the server IP and port. The command `importFile` will import and parse a delimited data file, returning an `H2OParsedData` object.

H2O supports a number of standard statistical models, such as GLM, K-means, and random forest classification. For example, to run GLM, call `h2o.glm` with the parsed data and parameters (response variable, error distribution) as arguments. (The operation will be done on the server associated with the data object).

Note that no actual data is stored in the workspace - R only saves the hex keys, which uniquely identify the data set, model, etc on the server. When the user makes a request, R queries the server via the REST API, which returns a JSON file with the relevant information that R then displays in the console.

Author(s)

Anqi Fu, Tom Kraljevic and Petr Maj, with contributions from the 0xdata team

Maintainer: Anqi Fu <anqi@0xdata.com>

References

- 0xdata Homepage
- H2O on Github

Examples

```
library(h2o)
h2o = new("H2OClient", ip = "localhost", port = 54321)
checkH2OClient(h2o)
prostate.hex = importURL(h2o, "https://raw.github.com/0xdata/h2o/master/smalldata/
logreg/prostate.csv", "prostate.hex")
summary(prostate.hex)
prostate.glm = h2o.glm(y = "CAPSULE", x = c("AGE","RACE","PSA","DCAPS"), data = prostate.hex,
family = "binomial", nfolds = 10, alpha = 0.5)
print(prostate.glm)
h2o.kmeans(data = prostate.hex, centers = 5)
```

---

checkH2OClient *Check if H2O is Running*

Description

Checks if H2O is running on the specified IP and port, and stops the program if it is not. Th method also compares the version number of the H2O program and the installed h2o R package. If there is a mismatch, it will warn the user.

Usage

```
checkH2OClient(object)
```

Arguments

object	An H2OClient object containing the IP address and port of the server running H2O.

Examples

```
myClient = new("H2OClient", ip = "localhost", port = 54321)
checkH2OClient(myClient)
```

---

h2o.getTree *Get Tree from Random Forest Model*

Description

Returns the depth and number of leaves of a particular tree in the random forest ensemble.

Usage

```
h2o.getTree(forest, k)
```

Arguments

forest	An H2ORForestModel object indicating the random forest model to examine.
k	The particular tree to retrieve. (Must be an integer between 1 and ntree).

See Also

h2o.randomForest, H2ORForestModel

Examples

```
iris.hex = importFile(h2o, "../smalldata/iris/iris.csv", "iris.hex")
iris.rf = h2o.randomForest(y = "4", data = iris.hex, ntree = 50)
h2o.getTree(iris.rf, k = 5)
```

---

h2o.glm *H2O: Generalized Linear Model*

Description

Fit a generalized linear model, specified by a response variable, a set of predictors, and a description of the error distribution.

Usage

```
h2o.glm(x, y, data, family, nfolds = 10, alpha = 0.5, lambda = 1e-05)
```

Arguments

x	A vector containing the names of the predictors in the model.
y	The name of the response variable in the model.
data	An H2OParsedData object containing the variables in the model.
family	A description of the error distribution and corresponding link function to be used in the model. Currently, Gaussian, binomial, Poisson, and gamma are supported.
nfolds	Number of folds for cross-validation. The default is 10.
alpha	The elastic-net mixing parameter, which must be in [0,1]. The penalty is defined to be $$P(\alpha,\beta) = (1-\alpha)/2||\beta||_2^2 + \alpha||\beta||_1 = \sum_j [(1-\alpha)/2\beta_j^2 + \alpha|\beta_j|]$$ so alpha=1 is the lasso penalty, while alpha=0 is the ridge penalty.
lambda	The shrinkage parameter, which multiples $P(\alpha,\beta)$ in the objective. The larger lambda is, the more the coefficients are shrunk toward zero (and each other).

Value

An object of class H2OGLMModel with slots key, data, and glm, where the last is a list of the following components:

coefficients	A named vector of the coefficients estimated in the model.
rank	The numeric rank of the fitted linear model.
family	The family of the error distribution.
deviance	The deviance of the fitted model.
aic	Akaike's Information Criterion for the final computed model.
null.deviance	The deviance for the null model.
iter	Number of algorithm iterations to compute the model.
df.residual	The residual degrees of freedom.
df.null	The residual degrees of freedom for the null model.
y	The response variable in the model.
x	A vector of the predictor variable(s) in the model.

Examples

```
# Run GLM of CAPSULE ~ AGE + RACE + PSA + DCAPS
prostate.hex = importURL(h2o, "https://raw.github.com/0xdata/h2o/master/smalldata/
   logreg/prostate.csv", "prostate.hex")
h2o.glm(y = "CAPSULE", x = c("AGE","RACE","PSA","DCAPS"), data = prostate.hex, family =
"binomial", nfolds = 10, alpha = 0.5)

# Run GLM of VOL ~ CAPSULE + AGE + RACE + PSA + GLEASON
myX = setdiff(colnames(prostate.hex), c("ID", "DPROS", "DCAPS", "VOL"))
h2o.glm(y = "VOL", x = myX, data = prostate.hex, family = "gaussian", nfolds = 5, alpha = 0.1)
```

h2o.kmeans *H2O: K-Means Clustering*

Description

Performs k-means clustering on a parsed data file.

Usage

```
h2o.kmeans(data, centers, cols = "", iter.max = 10)
```

Arguments

data	An H2OParsedData object containing the variables in the model.
centers	The number of clusters k.
cols	(Optional) A vector containing the names of the data columns on which k-means runs. If blank, k-means clustering will be run on the entire data set.
iter.max	The maximum number of iterations allowed.

Value

An object of class H2OKMeansModel with slots key, data, and km, where the last is a list of the following components:

centers	A matrix of cluster centers.
cluster	A H2OParsedData object containing the vector of integers (from 1 to k), which indicate the cluster to which each point is allocated.
size	The number of points in each cluster.
withinss	Vector of within-cluster sum of squares, with one component per cluster.
tot.withinss	Total within-cluster sum of squares, i.e., sum(withinss).

Examples

```
prostate.hex = importFile(h2o, "../smalldata/logreg/prostate.csv")
h2o.kmeans(data = prostate.hex, centers = 10, cols = c("AGE", "RACE", "VOL", "GLEASON"))

covtype.hex = importFile(h2o, "../smalldata/covtype/covtype.20k.data")
covtype.km = h2o.kmeans(data = covtype.hex, centers = 10, cols = c(1, 2, 3))
print(covtype.km)
```

h2o.randomForest *H2O: Random Forest*

Description

Performs random forest classification on a parsed data set.

Usage

```
h2o.randomForest(y, x_ignore = "", data, ntree, depth, classwt = as.numeric(NA))
```

Arguments

y	The name of the response variable. If the data does not contain a header, this is the column index. (This must be either an integer or a categorical variable).
x_ignore	(Optional) A vector containing the names of the predictor variables to ignore in building the random forest model. If blank, random forest will use all variables except y for classification.
data	An H2OParsedData object containing the variables in the model.
ntree	Number of trees to grow. (Must be a nonnegative integer).
depth	Maximum depth to grow the tree.
classwt	(Optional) Priors of the classes. Need not add up to one. If missing, defaults to all weights set at 1.0.

Details

Currently, only classification regression trees are supported, and there is no way to ignore predictor variables during growth of the tree.

Value

An object of class H2ORForestModel with slots key, data, and rf, where the last is a list of the following components:

type	The type of the tree, which at this point must be classification.
ntree	Number of trees grown.
oob_err	Out of bag error rate.
forest	A matrix giving the minimum, mean, and maximum of the tree depth and number of leaves.
confusion	Confusion matrix of the prediction.

Examples

```
# Assuming your working directory is h2o/R
iris.hex = importFile(h2o, "../smalldata/iris/iris.csv", "iris.hex")
h2o.randomForest(y = "4", data = iris.hex, ntree = 50, depth = 100,
   classwt = c("Iris-versicolor" = 20.0, "Iris-virginica" = 30.0))

prostate.hex = importFile(h2o, "../smalldata/logreg/prostate.csv", "prostate.hex")
h2o.randomForest(y = "CAPSULE", x_ignore = c("ID", "DPROS"), data = prostate.hex,
  ntree = 50, depth = 100)
```

H2OClient-class *Class* "H2OClient"

Description

An object representing the server/local machine on which H2O is running.

Objects from the Class

Objects can be created by calls of the form `new("H2OClient", ...)`

Slots

ip: Object of class `"character"` representing the IP address of the H2O server.

port: Object of class `"numeric"` representing the port number of the H2O server.

Methods

importFile `signature(object = "H2OClient", path = "character", key = "character", parse = "logical"`
...

importFolder `signature(object = "H2OClient", path = "character", parse = "logical")`:
...

importURL `signature(object = "H2OClient", path = "character", key = "character", parse = "logical`
...

show `signature(object = "H2OClient")`: ...

Examples

```
showClass("H2OClient")
localClient = new("H2OClient", ip = "localhost", port = 54321)
remoteClient = new("H2OClient", ip = "192.168.1.173", port = 54322)
```

H2OGLMModel-class *Class* "H2OGLMModel"

Description

A class for representing generalized linear models.

Objects from the Class

Objects can be created by calls of the form `new("H2OGLMModel", ...)`.

Slots

`key`: Object of class `"character"`, representing the unique hex key that identifies the model.

`data`: Object of class `H2OParsedData`, which is the input data used to build the model.

`glm`: Object of class `"list"` containing the following elements:

- `coefficients`: A named vector of the coefficients estimated in the model.
- `rank`: The numeric rank of the fitted linear model.
- `family`: The family of the error distribution.
- `deviance`: The deviance of the fitted model.
- `aic`: Akaike's Information Criterion for the final computed model.
- `null.deviance`: The deviance for the null model.
- `iter`: Number of algorithm iterations to compute the model.
- `df.residual`: The residual degrees of freedom.
- `df.null`: The residual degrees of freedom for the null model.
- `y`: The response variable in the model.
- `x`: A vector of the predictor variable(s) in the model.

Methods

show `signature(object = "H2OGLMModel")`: ...

Examples

```
showClass("H2OGLMModel")
```

H2OKMeansModel-class *Class* "H2OKMeansModel"

Description

An class for representing k-means models.

Objects from the Class

Objects can be created by calls of the form `new("H2OKMeansModel", ...)`.

Slots

`key`: Object of class `"character"`, representing the unique hex key that identifies the model.

`data`: Object of class `H2OParsedData`, which is the input data used to build the model.

`model`: Object of class `"list"` containing the following elements:

- `centers`: A matrix of cluster centers.
- `cluster`: A `H2OParsedData` object containing the vector of integers (from 1:k), which indicate the cluster to which each point is allocated.
- `size`: The number of points in each cluster.
- `withinss`: Vector of within-cluster sum of squares, with one component per cluster.
- `tot.withinss`: Total within-cluster sum of squares, i.e., `sum(withinss)`.

Methods

show signature(object = "H2OKMeansModel"): ...

Examples

```
showClass("H2OKMeansModel")
```

H2OParsedData-class *Class* "H2OParsedData"

Objects from the Class

Objects can be created by calls of the form new("H2OParsedData", ...).

Slots

h2o: Object of class "H2OClient", which is the client object that was passed into the function call.

key: Object of class "character", which is the hex key assigned to the imported data.

Methods

colnames signature(x = "H2OParsedData"): ...

h2o.glm signature(x = "character", y = "character", data = "H2OParsedData", family = "character"
...

h2o.glm signature(x = "character", y = "character", data = "H2OParsedData", family = "character"
...

h2o.kmeans signature(data = "H2OParsedData", centers = "numeric", cols = "ANY", iter.max = "ANY"
...

h2o.kmeans signature(data = "H2OParsedData", centers = "numeric", cols = "character", iter.max
...

h2o.randomForest signature(y = "character", data = "H2OParsedData", ntree = "numeric"):
...

show signature(object = "H2OParsedData"): ...

summary signature(object = "H2OParsedData"): ...

Examples

```
showClass("H2OParsedData")
```

H2ORawData-class *Class* "H2ORawData"

Objects from the Class

Objects can be created by calls of the form `new("H2ORawData", ...)`.

Slots

h2o: Object of class "H2OClient" ~~

key: Object of class "character" ~~

Methods

parseRaw `signature(data = "H2ORawData", key = "character")`: ...

parseRaw `signature(data = "H2ORawData", key = "missing")`: ...

show `signature(object = "H2ORawData")`: ...

Examples

```
showClass("H2ORawData")
```

H2ORForestModel-class *Class* "H2ORForestModel"

Description

A class for representing random forest ensembles.

Objects from the Class

Objects can be created by calls of the form `new("H2ORForestModel", ...)`.

Slots

key: Object of class "character", representing the unique hex key that identifies the model.

data: Object of class H2OParsedData, which is the input data used to build the model.

model: Object of class "list" containing the following elements:

- type: The type of the tree, which at this point must be classification.
- ntree: Number of trees grown.
- oob_err: Out of bag error rate.
- forest: A matrix giving the minimum, mean, and maximum of the tree depth and number of leaves.
- confusion: Confusion matrix of the prediction.

Methods

h2o.getTree `signature(forest = "H2ORForestModel", k = "numeric")`: ...

show `signature(object = "H2ORForestModel")`: ...

Examples

```
showClass("H2ORForestModel")
```

`importFile` *Import Local Data File*

Description

Imports a file from the local path and parses it, returning an object containing the identifying hex key.

Usage

```
importFile(object, path, key = "", parse = TRUE)
```

Arguments

`object`	An `H2OClient` object containing the IP address and port of the server running H2O.
`path`	The path of the file to be imported. Each row of data appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory.
`key`	(Optional) The unique hex key assigned to the imported file. If none is given, a key will automatically be generated based on the file path.
`parse`	(Optional) A logical value indicating whether the file should be parsed after import.

Details

WARNING: In H2O, import is lazy! Do not modify the data on hard disk until after parsing is complete.

Value

If `parse = TRUE`, the function returns an object of class `H2OParsedData`, otherwise it returns an object of class `H2ORawData`.

Examples

```
h2o = new("H2OClient", ip="localhost", port=54321)
benign.hex = importFile(h2o, "../smalldata/logreg/benign.csv", "benign.hex")
summary(benign.hex)
```

importFolder *Import Local Directory*

Description

Imports all the files in the local directory and parses them, returning a list of objects containing the identifying hex keys.

Usage

```
importFolder(object, path, parse = TRUE)
```

Arguments

object	An H2OClient object containing the IP address and port of the server running H2O.
path	The path of the folder directory to be imported. Each row of data appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory.

Details

WARNING: In H2O, import is lazy! Do not modify the data files on hard disk until after parsing is complete.

Value

If parse = TRUE, the function returns a list of objects of class H2OParsedData, otherwise it returns a list of objects of class H2ORawData.

Examples

```
h2o = new("H2OClient", ip="localhost", port=54321)
glm_test.hex = importFolder(h2o, "../smalldata/glm_test")
for(i in 1:length(glm_test.hex))
  print(summary(glm_test.hex[[i]]))
```

importHDFS *Import from HDFS*

Description

Imports a HDFS file or set of files in a directory and parses them, returning a list of objects containing the identifying hex keys.

Usage

```
importHDFS(object, path, parse = TRUE)
```

Arguments

object	An H2OClient object containing the IP address and port of the server running H2O.
path	The path of the folder directory to be imported. Each row of data appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory.
parse	(Optional) A logical value indicating whether the file should be parsed after import.

Details

WARNING: In H2O, import is lazy! Do not modify the data files on hard disk until after parsing is complete.

Value

If `parse = TRUE`, the function returns a list of objects of class H2OParsedData, otherwise it returns a list of objects of class H2ORawData.

See Also

importFolder, importFile, importURL

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
{
  }
```

---

importURL *Import Data from URL*

Description

Imports a file from the URL and parses it, returning an object containing the identifying hex key.

Usage

```
importURL(object, path, key = "", parse = TRUE)
```

Arguments

object	An H2OClient object containing the IP address and port of the server running H2O.
path	The complete URL of the file to be imported. Each row of data appears as one line of the file.
key	(Optional) The unique hex key assigned to the imported file. If none is given, a key will automatically be generated based on the URL path.
parse	(Optional) A logical value indicating whether the file should be parsed after import.

Details

WARNING: In H2O, import is lazy! Do not modify the data on hard disk until after parsing is complete.

Value

If parse = TRUE, the function returns an object of class H2OParsedData, otherwise it returns an object of class H2ORawData.

Examples

```
h2o = new("H2OClient", ip="localhost", port=54321)
prostate.hex = importURL(h2o, "https://raw.github.com/0xdata/h2o/master/smalldata/
   logreg/prostate.csv", "prostate.hex")
summary(prostate.hex)
```

---

parseRaw *Parse Raw Data File*

Description

Parses a raw data file, returning an object containing the identifying hex key.

Usage

```
parseRaw(data, key = "")
```

Arguments

data	An H2ORawData object to be parsed.
key	(Optional) The hex key assigned to the parsed file.

Details

After the raw data file is parsed, it will be automatically deleted from the H2O server.

Examples

```
h2o = new("H2OClient", ip = "localhost", port = 54321)
benign.raw = importFile(h2o, path="../smalldata/logreg/benign.csv", parse=FALSE)
benign.hex = parseRaw(benign.hex, "benign.hex")
```