Package ‘xgboost’

December 22, 2014

Type Package

Title eXtreme Gradient Boosting

Version 0.3-2

Date 2014-08-23

Author Tianqi Chen <`tianqi.tchen@gmail.com`>, Tong He <`hetong007@gmail.com`>

Maintainer Tong He <`hetong007@gmail.com`>

Description This package is a R wrapper of xgboost, which is short for eXtreme Gradient Boosting. It is an efficient and scalable implementation of gradient boosting framework. The package includes efficient linear model solver and tree learning algorithms. The package can automatically do parallel computation with OpenMP, and it can be more than 10 times faster than existing gradient boosting packages such as gbm. It supports various objective functions, including regression, classification and ranking. The package is made to be extensible, so that users are also allowed to define their own objectives easily.

License Apache License (== 2.0) | file LICENSE

URL `https://github.com/tqchen/xgboost`

BugReports `https://github.com/tqchen/xgboost/issues`

Depends R (>= 2.10)

Imports Matrix (>= 1.1-0), methods

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-09-07 21:54:44

R topics documented:

`agaricus.test` *Test part from Mushroom Data Set*

Description

This data set is originally from the Mushroom data set, UCI Machine Learning Repository.

Usage

```
data(agaricus.test)
```

Format

A list containing a label vector, and a dgCMatrix object with 1611 rows and 127 variables

Details

This data set includes the following fields:

- `label` the label for each record
- `data` a sparse Matrix of `dgCMatrix` class, with 127 columns.

References

https://archive.ics.uci.edu/ml/datasets/Mushroom

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

`agaricus.train` *Training part from Mushroom Data Set*

Description

This data set is originally from the Mushroom data set, UCI Machine Learning Repository.

Usage

```
data(agaricus.train)
```

Format

A list containing a label vector, and a dgCMatrix object with 6513 rows and 127 variables

Details

This data set includes the following fields:

- `label` the label for each record
- `data` a sparse Matrix of `dgCMatrix` class, with 127 columns.

References

https://archive.ics.uci.edu/ml/datasets/Mushroom

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

`getinfo` *Get information of an xgb.DMatrix object*

Description

Get information of an xgb.DMatrix object

Usage

```
getinfo(object, ...)

## S4 method for signature 'xgb.DMatrix'
getinfo(object, name)
```

Arguments

`object`	Object of class "xgb.DMatrix"
`name`	the name of the field to get
`...`	other parameters

Examples

```
data(agaricus.train, package='xgboost')
train <- agaricus.train
dtrain <- xgb.DMatrix(train$data, label=train$label)
labels <- getinfo(dtrain, 'label')
setinfo(dtrain, 'label', 1-labels)
labels2 <- getinfo(dtrain, 'label')
stopifnot(all(labels2 == 1-labels))
```

`predict,xgb.Booster-method` *Predict method for eXtreme Gradient Boosting model*

Description

Predicted values based on xgboost model object.

Usage

```
## S4 method for signature 'xgb.Booster'
predict(object, newdata, outputmargin = FALSE,
  ntreelimit = NULL)
```

Arguments

`object`	Object of class "xgb.Boost"
`newdata`	takes `matrix`, `dgCMatrix`, local data file or `xgb.DMatrix`.
`outputmargin`	whether the prediction should be shown in the original value of sum of functions, when outputmargin=TRUE, the prediction is untransformed margin value. In logistic regression, outputmargin=T will output value before logistic transformation.
`ntreelimit`	limit number of trees used in prediction, this parameter is only valid for gbtree, but not for gblinear. set it to be value bigger than 0. It will use all trees by default.

Examples

```
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
train <- agaricus.train
test <- agaricus.test
bst <- xgboost(data = train$data, label = train$label, max.depth = 2,
               eta = 1, nround = 2,objective = "binary:logistic")
pred <- predict(bst, test$data)
```

---

setinfo *Set information of an xgb.DMatrix object*

Description

Set information of an xgb.DMatrix object

Usage

```
setinfo(object, ...)

## S4 method for signature 'xgb.DMatrix'
setinfo(object, name, info)
```

Arguments

`object`	Object of class "xgb.DMatrix"
`name`	the name of the field to get
`info`	the specific field of information to set
`...`	other parameters

Examples

```
data(agaricus.train, package='xgboost')
train <- agaricus.train
dtrain <- xgb.DMatrix(train$data, label=train$label)
labels <- getinfo(dtrain, 'label')
setinfo(dtrain, 'label', 1-labels)
labels2 <- getinfo(dtrain, 'label')
stopifnot(all(labels2 == 1-labels))
```

slice — *Get a new DMatrix containing the specified rows of orginal xgb.DMatrix object*

Description

Get a new DMatrix containing the specified rows of orginal xgb.DMatrix object

Usage

```
slice(object, ...)

## S4 method for signature 'xgb.DMatrix'
slice(object, idxset, ...)
```

Arguments

`object`	Object of class "xgb.DMatrix"
`idxset`	a integer vector of indices of rows needed
`...`	other parameters

Examples

```
data(agaricus.train, package='xgboost')
train <- agaricus.train
dtrain <- xgb.DMatrix(train$data, label=train$label)
dsub <- slice(dtrain, 1:3)
```

xgb.cv — *Cross Validation*

Description

The cross valudation function of xgboost

Usage

```
xgb.cv(params = list(), data, nrounds, nfold, label = NULL, showsd = TRUE,
  metrics = list(), obj = NULL, feval = NULL, ...)
```

Arguments

`params`	the list of parameters. Commonly used ones are: • `objective` objective function, common ones are – `reg:linear` linear regression – `binary:logistic` logistic regression for classification • `eta` step size of each boosting step • `max.depth` maximum depth of the tree • `nthread` number of thread used in training, if not set, all threads are used See `https://github.com/tqchen/xgboost/wiki/Parameters` for further details. See also demo/ for walkthrough example in R.
`data`	takes an `xgb.DMatrix` as the input.
`nrounds`	the max number of iterations
`nfold`	number of folds used
`label`	option field, when data is Matrix
`showsd`	boolean, whether show standard deviation of cross validation
`metrics,`	list of evaluation metrics to be used in corss validation, when it is not specified, the evaluation metric is chosen according to objective function. Possible options are: • `error` binary classification error rate • `rmse` Rooted mean square error • `logloss` negative log-likelihood function • `auc` Area under curve • `merror` Exact matching error, used to evaluate multi-class classification
`obj`	customized objective function. Returns gradient and second order gradient with given prediction and dtrain,
`feval`	custimized evaluation function. Returns `list(metric='metric-name', value='metric-value')` with given prediction and dtrain,
`...`	other parameters to pass to `params`.

Details

This is the cross validation function for xgboost

Parallelization is automatically enabled if OpenMP is present. Number of threads can also be manually specified via "nthread" parameter.

This function only accepts an `xgb.DMatrix` object as the input.

Examples

```
data(agaricus.train, package='xgboost')
dtrain <- xgb.DMatrix(agaricus.train$data, label = agaricus.train$label)
history <- xgb.cv(data = dtrain, nround=3, nfold = 5, metrics=list("rmse","auc"),
                  "max.depth"=3, "eta"=1, "objective"="binary:logistic")
```

xgb.DMatrix *Contruct xgb.DMatrix object*

Description

Contruct xgb.DMatrix object from dense matrix, sparse matrix or local file.

Usage

```
xgb.DMatrix(data, info = list(), missing = 0, ...)
```

Arguments

`data`	a `matrix` object, a `dgCMatrix` object or a character indicating the data file.
`info`	a list of information of the xgb.DMatrix object
`missing`	Missing is only used when input is dense matrix, pick a float
`...`	other information to pass to `info`.

Examples

```
data(agaricus.train, package='xgboost')
train <- agaricus.train
dtrain <- xgb.DMatrix(train$data, label=train$label)
xgb.DMatrix.save(dtrain, 'xgb.DMatrix.data')
dtrain <- xgb.DMatrix('xgb.DMatrix.data')
```

xgb.DMatrix.save *Save xgb.DMatrix object to binary file*

Description

Save xgb.DMatrix object to binary file

Usage

```
xgb.DMatrix.save(DMatrix, fname)
```

Arguments

`DMatrix`	the DMatrix object
`fname`	the name of the binary file.

Examples

```
data(agaricus.train, package='xgboost')
train <- agaricus.train
dtrain <- xgb.DMatrix(train$data, label=train$label)
xgb.DMatrix.save(dtrain, 'xgb.DMatrix.data')
dtrain <- xgb.DMatrix('xgb.DMatrix.data')
```

`xgb.dump` *Save xgboost model to text file*

Description

Save a xgboost model to text file. Could be parsed later.

Usage

```
xgb.dump(model, fname, fmap = "")
```

Arguments

`model`	the model object.
`fname`	the name of the binary file.
`fmap`	feature map file representing the type of feature. Detailed description could be found at `https://github.com/tqchen/xgboost/wiki/Binary-Classification#dump-model`. See demo/ for walkthrough example in R, and `https://github.com/tqchen/xgboost/blob/master/demo/data/featmap.txt` for example Format.

Examples

```
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
train <- agaricus.train
test <- agaricus.test
bst <- xgboost(data = train$data, label = train$label, max.depth = 2,
               eta = 1, nround = 2,objective = "binary:logistic")
xgb.dump(bst, 'xgb.model.dump')
```

`xgb.load` *Load xgboost model from binary file*

Description

Load xgboost model from the binary model file

Usage

```
xgb.load(modelfile)
```

Arguments

`modelfile`	the name of the binary file.

Examples

```
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
train <- agaricus.train
test <- agaricus.test
bst <- xgboost(data = train$data, label = train$label, max.depth = 2,
               eta = 1, nround = 2,objective = "binary:logistic")
xgb.save(bst, 'xgb.model')
bst <- xgb.load('xgb.model')
pred <- predict(bst, test$data)
```

`xgb.save` *Save xgboost model to binary file*

Description

Save xgboost model from xgboost or xgb.train

Usage

```
xgb.save(model, fname)
```

Arguments

`model`	the model object.
`fname`	the name of the binary file.

Examples

```
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
train <- agaricus.train
test <- agaricus.test
bst <- xgboost(data = train$data, label = train$label, max.depth = 2,
               eta = 1, nround = 2,objective = "binary:logistic")
xgb.save(bst, 'xgb.model')
bst <- xgb.load('xgb.model')
pred <- predict(bst, test$data)
```

`xgb.train` *eXtreme Gradient Boosting Training*

Description

The training function of xgboost

Usage

```
xgb.train(params = list(), data, nrounds, watchlist = list(), obj = NULL,
  feval = NULL, verbose = 1, ...)
```

Arguments

`params`	the list of parameters. Commonly used ones are: • `objective` objective function, common ones are – `reg:linear` linear regression – `binary:logistic` logistic regression for classification • `eta` step size of each boosting step • `max.depth` maximum depth of the tree • `nthread` number of thread used in training, if not set, all threads are used See `https://github.com/tqchen/xgboost/wiki/Parameters` for further details. See also demo/ for walkthrough example in R.
`data`	takes an `xgb.DMatrix` as the input.
`nrounds`	the max number of iterations
`watchlist`	what information should be printed when `verbose=1` or `verbose=2`. Watchlist is used to specify validation set monitoring during training. For example user can specify watchlist=list(validation1=mat1, validation2=mat2) to watch the performance of each round's model on mat1 and mat2
`obj`	customized objective function. Returns gradient and second order gradient with given prediction and dtrain,
`feval`	custimized evaluation function. Returns `list(metric='metric-name', value='metric-value')` with given prediction and dtrain,
`verbose`	If 0, xgboost will stay silent. If 1, xgboost will print information of performance. If 2, xgboost will print information of both
`...`	other parameters to pass to `params`.

Details

This is the training function for xgboost.

Parallelization is automatically enabled if OpenMP is present. Number of threads can also be manually specified via "nthread" parameter.

This function only accepts an `xgb.DMatrix` object as the input. It supports advanced features such as watchlist, customized objective function, therefore it is more flexible than `xgboost`.

Examples

```
data(agaricus.train, package='xgboost')
dtrain <- xgb.DMatrix(agaricus.train$data, label = agaricus.train$label)
dtest <- dtrain
watchlist <- list(eval = dtest, train = dtrain)
param <- list(max.depth = 2, eta = 1, silent = 1)
logregobj <- function(preds, dtrain) {
   labels <- getinfo(dtrain, "label")
   preds <- 1/(1 + exp(-preds))
   grad <- preds - labels
   hess <- preds * (1 - preds)
   return(list(grad = grad, hess = hess))
}
evalerror <- function(preds, dtrain) {
  labels <- getinfo(dtrain, "label")
  err <- as.numeric(sum(labels != (preds > 0)))/length(labels)
  return(list(metric = "error", value = err))
}
bst <- xgb.train(param, dtrain, nround = 2, watchlist, logregobj, evalerror)
```

`xgboost` *eXtreme Gradient Boosting (Tree) library*

Description

A simple interface for xgboost in R

Usage

```
xgboost(data = NULL, label = NULL, params = list(), nrounds,
  verbose = 1, ...)
```

Arguments

`data`	takes `matrix`, `dgCMatrix`, local data file or `xgb.DMatrix`.
`label`	the response variable. User should not set this field,
`params`	the list of parameters. Commonly used ones are: • `objective` objective function, common ones are – `reg:linear` linear regression

    - `binary:logistic` logistic regression for classification
- `eta` step size of each boosting step
- `max.depth` maximum depth of the tree
- `nthread` number of thread used in training, if not set, all threads are used

See https://github.com/tqchen/xgboost/wiki/Parameters for further details. See also demo/ for walkthrough example in R.

`nrounds`	the max number of iterations
`verbose`	If 0, xgboost will stay silent. If 1, xgboost will print information of performance. If 2, xgboost will print information of both performance and construction progress information
`...`	other parameters to pass to `params`.

Details

This is the modeling function for xgboost.

Parallelization is automatically enabled if OpenMP is present. Number of threads can also be manually specified via "nthread" parameter

Examples

```
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
train <- agaricus.train
test <- agaricus.test
bst <- xgboost(data = train$data, label = train$label, max.depth = 2,
               eta = 1, nround = 2,objective = "binary:logistic")
pred <- predict(bst, test$data)
```

Index