Package ‘gcForest’

October 19, 2018

Type Package
Title Deep Forest Model
Version 0.2.7
Author Xu Jing [cre]
Maintainer Xu Jing <274762204@qq.com>
License GPL (>= 2)
SystemRequirements Python (>= 3.5.0)
Encoding UTF-8
LazyData true
URL https://github.com/DataXujing/gcForest_r
BugReports https://github.com/DataXujing/gcForest_r/issues
RoxygenNote 6.0.1
Depends R (>= 3.4.0)
Imports reticulate,pkgdown,crayon,cli,utils
Suggests rmarkdown, knitr
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2018-10-19 14:00:03 UTC


**R topics documented:**

- `gcForest-package` ................................................................. 2
- `gcdata` .................................................................................... 3
- `gcforest` .................................................................................. 4
- `model_load` .............................................................................. 6
- `model_save` .............................................................................. 7
- `req_py` ................................................................................. 9

---

**Description**


**Author(s)**

Xu Jing

**See Also**


[3] [https://github.com/pylablanche/gcForest](https://github.com/pylablanche/gcForest)

**Examples**

```r
# ======== Model train========

have_numpy <- reticulate::py_module_available("numpy")
have_sklearn <- reticulate::py_module_available("sklearn")

if(have_numpy & have_sklearn){
  library(gcForest)
  # req_py()

  sk <- NULL

  .onLoad <- function(libname, pkgname) {
```
```r
sk <- reticulate::import("sklearn", delay_load = TRUE)
}
sk <- reticulate::import("sklearn", delay_load = TRUE)
train_test_split <- sk$model_selection$train_test_split
data <- sk$datasets$load_iris
iris <- data()
X = iris$data
y = iris$target
data_split = train_test_split(X, y, test_size=0.33)

X_tr <- data_split[[1]]
X_te <- data_split[[2]]
y_tr <- data_split[[3]]
y_te <- data_split[[4]]

gcforest_m <- gcforest(shape_1x=4L, window=2L, tolerance=0.0)
gcforest_m$fit(X_tr, y_tr)
gcf_model <- model_save(gcforest_m, 'gcforest_model.model')

gcf <- model_load('gcforest_model.model')
gcf$predict(X_te)

# learn more from gcForest package tutorial
utils::vignette('gcForest-docs')
)

```

---

**R Data Transform to Python Data**

**Description**

A function to transform R data structure to Python data structure, which based on the reticulate package.

**Usage**

```r
gcdata(x)
```

**Arguments**

- **x**
  
The R project like data.frame, vector, array etc..

**Author(s)**

Xu Jing
Examples

```r
have_numpy <- reticulate::py_module_available("numpy")
have_sklearn <- reticulate::py_module_available("sklearn")

if(have_numpy && have_sklearn){
  library(gcForest)
  req_py()
  r_dat <- data.frame('x1'=c(1L,2L,3L),'x2'=c(2L,3L,4L))
  py_dat <- gcdata(r_dat)
  class(py_dat)
  r_vec <- c('a','b','c')
  py_vec <- gcdata(r_vec)
  class(py_vec)
} else{
  print('You should have the Python testing environment!')
}
```

gcforest  

**R for Deep Forest Model (gcForest)**

Description


Usage

gcforest(shape_1x=NULL, n_mgsRtree=30L, window=NULL, stride=1L,
  cascade_test_size=0.2, n_cascadeRF=2L, n_cascadeRtree=101L,
  cascade_layer=Inf, min_samples_mgs=0.1, min_samples_cascade=0.05,
  tolerance=0.0)

Arguments

- **shape_1x**: int or tuple list or np.array (default=None)shape of a single sample element [n_lines, n_cols]. Required when calling mg_scanning!For sequence data a single int can be given.
- **n_mgsRtree**: int (default=30) Number of trees in a Random Forest during Multi Grain Scanning.
- **window**: int (default=None)List of window sizes to use during Multi Grain Scanning. If 'None' no slicing will be done.
- **stride**: int (default=1)Step used when slicing the data.
cascade_test_size
float or int (default=0.2) Split fraction or absolute number for cascade training set splitting.

n_cascadeRF
int (default=2) Number of Random Forests in a cascade layer. For each pseudo Random Forest a complete Random Forest is created, hence the total number of Random Forests in a layer will be 2*n_cascadeRF.

n_cascadeRFtreen_cascadeRFtreen_cascadeRFtree
int (default=101) Number of trees in a single Random Forest in a cascade layer.

cascade_layer
int (default=np.inf) Maximum number of cascade layers allowed. Useful to limit the contraction of the cascade.

min_samples_mgs
float or int (default=0.1) Minimum number of samples in a node to perform a split during the training of Multi-Grain Scanning Random Forest. If int number_of_samples = int. If float, min_samples represents the fraction of the initial n_samples to consider.

min_samples_cascademin_samples_cascade
float or int (default=0.1) Minimum number of samples in a node to perform a split during the training of Cascade Random Forest. If int number_of_samples = int. If float, min_samples represents the fraction of the initial n_samples to consider.

tolerance
tolerance
float (default=0.0) Accuracy tolerance for the cascade growth. If the improvement in accuracy is not better than the tolerance the construction is stopped.

Details
gcForest provides several important function interfaces, just like the style of Python sklearn.

1. fit(X, y) Training the gcForest on input data X and associated target y;
2. predict(X) Predict the class of unknown samples X;
3. predict_proba(X) Predict the class probabilities of unknown samples X;
4. mg_scanning(X, y=None) Performs a Multi Grain Scanning on input data;
5. window_slicing_pred_prob(X, window, shape_1X, y=None) Performs a window slicing of the input data and send them through Random Forests. If target values ‘y’ are provided sliced data are then used to train the Random Forests;
6. cascade_forest(X, y=None) Perform (or train if ‘y’ is not None) a cascade forest estimator;

Author(s)
Xu Jing

Examples

```r
have_numpy <- reticulate::py_module_available("numpy")
have_sklearn <- reticulate::py_module_available("sklearn")

if(have_numpy & have_sklearn){
```
library(gcForest)
req_py()

sk <- NULL

.onLoad <- function(libname, pkgname) {
  sk <- reticulate::import("sklearn", delay_load = TRUE)
}

sk <- reticulate::import("sklearn", delay_load = TRUE)
train_test_split <- sk$model_selection$train_test_split

data <- sk$datasets$load_iris
iris <- data()
X = iris$data
y = iris$target
data_split = train_test_split(X, y, test_size=0.33)

X_tr <- data_split[[1]]
X_te <- data_split[[2]]
y_tr <- data_split[[3]]
y_te <- data_split[[4]]

gcforest_m <- gcforest(shape_1X=4L, window=2L, tolerance=0.0)
gcforest_m$fit(X_tr, y_tr)

pred_X = gcforest_m$predict(X_te)
print(pred_X)
} else {
  print('You should have the Python testing environment!')
}

---

### model_load
gcForest Model Persistence Function

#### Description
It is a sklearn APIs to save your training model, and load it to predict, now you can use R to callback.
see also model_save

#### Usage

model_load(path)

#### Arguments

path The path to save model(see also model_save.)
Author(s)
Xu Jing

Examples

```r
have_numpy <- reticulate::py_module_available("numpy")
have_sklearn <- reticulate::py_module_available("sklearn")

if(have_numpy && have_sklearn){
  library(gcForest)
  req_py()

  sk <- NULL

  .onLoad <- function(libname, pkgname) {
    sk <<- reticulate::import("sklearn", delay_load = TRUE)
  }
  sk <<- reticulate::import("sklearn", delay_load = TRUE)
  train_test_split <- sk$model_selection$train_test_split

  data <- sk$datasets$load_iris
  iris <- data()
  X = iris$data
  y = iris$target
  data_split = train_test_split(X, y, test_size=0.33)

  X_tr <- data_split[[1]]
  X_te <- data_split[[2]]
  y_tr <- data_split[[3]]
  y_te <- data_split[[4]]

  gcforest_m <- gcforest(shape_1X=4L, window=2L, tolerance=0.0)
  gcforest_m$fit(X_tr, y_tr)
  gcforest_model <- model_save(gcforest_m,'gcforest_model.model')

  gcf <- model_load('gcforest_model.model')
  gcf$predict(X_te)
}
else{
  print('You should have the Python testing environment!')
}
```

---

model_save gcForest Model Persistence Function
Description

It is a sklearn APIs to save your training model, and load it to predict, now you can use R to callback.
see also model_load

Usage

model_save(model, path)

Arguments

model The train model, like gcforest (see also gcforest).
path The path to save model.

Author(s)

Xu Jing

Examples

have_numpy <- reticulate::py_module_available("numpy")
have_sklearn <- reticulate::py_module_available("sklearn")

if(have_numpy && have_sklearn){
  library(gcForest)
  req_py()

  sk <- NULL
  .onLoad <- function(libname, pkname) {
    sk <- reticulate::import("sklearn", delay_load = TRUE)
  }
  sk <- reticulate::import("sklearn", delay_load = TRUE)
  train_test_split <- sk$model_selection$train_test_split

  data <- sk$datasets$load_iris
  iris <- data()
  X = iris$data
  y = iris$target
  data_split = train_test_split(X, y, test_size=0.33)

  X_tr <- data_split[[1]]
  X_te <- data_split[[2]]
  y_tr <- data_split[[3]]
  y_te <- data_split[[4]]

  gcforest_m <- gcforest(shape_1x=4L, window=2L, tolerance=0.0)
  gcforest_m$fit(X_tr, y_tr)
  gcf_model <- model_save(gcforest_m,'gcforest_model.model')

  gcf <- model_load('gcforest_model.model')
req_py

gcf$predict(X_te)

}else{
    print('You should have the Python testing environment!')
}

---

req_py | Detect Python Module

**Description**

A function to detect Python module.

**Usage**

req_py()

**Author(s)**

Xu Jing
Index

gcdata, 3
gcforest, 4, 8
gcForest-package, 2

model_load, 6, 8
model_save, 6, 7

req_py, 9