Package: mlsauce (via r-universe)

August 27, 2024

Title Miscellaneous Statistical/Machine Learning stuff

Version 0.18.1

Author T. Moudiki

Maintainer T. Moudiki <thierry.moudiki@gmail.com>

Description Miscellaneous Statistical/Machine Learning stuff.

License BSD_3_clause Clear + file LICENSE

Imports memoise

Depends reticulate

Suggests Rcpp

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

Repository https://techtonique.r-universe.dev

RemoteUrl https://github.com/Techtonique/mlsauce_r

RemoteRef HEAD

RemoteSha 60ceb48dd798ade9978610b040299e704a74ef02

Contents

AdaOpt *AdaOpt classifier*

Description

AdaOpt classifier

Usage

```
AdaOpt(
  n_iterations = 50L,
  learning_rate = 0.3,
  reg_lambda = 0.1,
  reg_alpha = 0.5,
  eta = 0.01,
  gamma = 0.01,
  k = 3L,
  tolerance = 0,
  n_clusters = 0,
  batch_size = 100L,
  row_sample = 1,
  type_dist = "euclidean-f",
  cache = TRUE,
  seed = 123L
)
```

Arguments

n_iterations	number of iterations of the optimizer at training time
learning_rate	controls the speed of the optimizer at training time
reg_lambda	L2 regularization parameter for successive errors in the optimizer (at training time)
reg_alpha	L1 regularization parameter for successive errors in the optimizer (at training time)
eta	controls the slope in gradient descent (at training time)
gamma	controls the step size in gradient descent (at training time)
k	number of nearest neighbors selected at test time for classification
tolerance	controls early stopping in gradient descent (at training time)
n_clusters	number of clusters, if MiniBatch k-means is used at test time (for faster prediction)
batch_size	size of the batch, if MiniBatch k-means is used at test time (for faster prediction)
row_sample	percentage of rows chosen from training set (by stratified subsampling, for faster prediction)

type_dist	distance used for finding the nearest neighbors; currently euclidean-f (euclidean distances calculated as whole), euclidean (euclidean distances calculated row by row), cosine (cosine distance)
cache	if the nearest neighbors are cached or not, for faster retrieval in subsequent calls
seed	reproducibility seed for initial weak learner and clustering

Value

An object of class AdaOpt

Examples

```
## Not run:
library(datasets)

X <- as.matrix(iris[, 1:4])
y <- as.integer(iris[, 5]) - 1L

n <- dim(X)[1]
p <- dim(X)[2]
set.seed(21341)
train_index <- sample(x = 1:n, size = floor(0.8*n), replace = TRUE)
test_index <- -train_index
X_train <- as.matrix(iris[train_index, 1:4])
y_train <- as.integer(iris[train_index, 5]) - 1L
X_test <- as.matrix(iris[test_index, 1:4])
y_test <- as.integer(iris[test_index, 5]) - 1L


obj <- mlsauce::AdaOpt()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))
## End(Not run)
```

---

download *Download datasets*

Description

Download datasets

Usage

```
download(
  pkgname = "MASS",
  dataset = "Boston",
  source = "https://cran.r-universe.dev/"
)
```

Arguments

pkgname	a string; R package name
dataset	a string; dataset name
source	a string; package location (address)

Value

A data frame

Examples

```
df <- mlsauce::download(pkgname="MASS", dataset="Boston", source="https://cran.r-universe.dev/")
print(df)
```

LassoRegressor *Lasso regressor*

Description

Lasso regressor

Usage

```
LassoRegressor(reg_lambda = 0.1, max_iter = 10L, tol = 0.001)
```

Arguments

reg_lambda	L1 regularization parameter
max_iter	number of iterations of lasso shooting algorithm.
tol	tolerance for convergence of lasso shooting algorithm.

Value

An object of class Lasso

Examples

```
## Not run:
library(datasets)

X <- as.matrix(datasets::mtcars[, -1])
y <- as.integer(datasets::mtcars[, 1])

n <- dim(X)[1]
p <- dim(X)[2]
set.seed(21341)
train_index <- sample(x = 1:n, size = floor(0.8*n), replace = TRUE)
test_index <- -train_index
X_train <- as.matrix(X[train_index, ])
y_train <- as.double(y[train_index])
X_test <- as.matrix(X[test_index, ])
y_test <- as.double(y[test_index])

obj <- mlsauce::LassoRegressor()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))
## End(Not run)
```

LSBoostClassifier *LSBoost classifier*

Description

LSBoost classifier

Usage

```
LSBoostClassifier(
  n_estimators = 100L,
  learning_rate = 0.1,
  n_hidden_features = 5L,
  reg_lambda = 0.1,
  row_sample = 1,
  col_sample = 1,
  dropout = 0,
  tolerance = 1e-04,
  direct_link = 1L,
  verbose = 1L,
  seed = 123L,
```

```
  solver = c("ridge", "lasso"),
  activation = "relu"
)
```

Arguments

n_estimators:	int, number of boosting iterations.
learning_rate:	float, controls the learning speed at training time.
n_hidden_features:	int
number	of nodes in successive hidden layers.
reg_lambda:	float, L2 regularization parameter for successive errors in the optimizer (at training time).
row_sample:	float, percentage of rows chosen from the training set.
col_sample:	float, percentage of columns chosen from the training set.
dropout:	float, percentage of nodes dropped from the training set.
tolerance:	float, controls early stopping in gradient descent (at training time).
direct_link:	bool, indicates whether the original features are included (True) in model's fitting or not (False).
verbose:	int, progress bar (yes = 1) or not (no = 0) (currently).
seed:	int, reproducibility seed for nodes_sim=='uniform', clustering and dropout.
solver:	str, type of 'weak' learner; currently in ('ridge', 'lasso')
activation:	str, activation function: currently 'relu', 'relu6', 'sigmoid', 'tanh'

Value

An object of class LSBoostClassifier

Examples

```
library(datasets)

X <- as.matrix(iris[, 1:4])
y <- as.integer(iris[, 5]) - 1L

n <- dim(X)[1]
p <- dim(X)[2]
set.seed(21341)
train_index <- sample(x = 1:n, size = floor(0.8*n), replace = TRUE)
test_index <- -train_index
X_train <- as.matrix(X[train_index, ])
y_train <- as.integer(y[train_index])
X_test <- as.matrix(X[test_index, ])
y_test <- as.integer(y[test_index])

## Not run:
obj <- mlsauce::LSBoostClassifier()
```

```
print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))
## End(Not run)
```

`LSBoostRegressor` *LSBoost Regressor*

Description

LSBoost Regressor

Usage

```
LSBoostRegressor(
  n_estimators = 100L,
  learning_rate = 0.1,
  n_hidden_features = 5L,
  reg_lambda = 0.1,
  row_sample = 1,
  col_sample = 1,
  dropout = 0,
  tolerance = 1e-04,
  direct_link = 1L,
  verbose = 1L,
  seed = 123L,
  solver = c("ridge", "lasso"),
  activation = "relu"
)
```

Arguments

`n_estimators:`	int, number of boosting iterations.
`learning_rate:`	float, controls the learning speed at training time.
`n_hidden_features:`	int
`number`	of nodes in successive hidden layers.
`reg_lambda:`	float, L2 regularization parameter for successive errors in the optimizer (at training time).
`row_sample:`	float, percentage of rows chosen from the training set.
`col_sample:`	float, percentage of columns chosen from the training set.
`dropout:`	float, percentage of nodes dropped from the training set.

tolerance:	float, controls early stopping in gradient descent (at training time).
direct_link:	bool, indicates whether the original features are included (True) in model's fitting or not (False).
verbose:	int, progress bar (yes = 1) or not (no = 0) (currently).
seed:	int, reproducibility seed for nodes_sim=='uniform', clustering and dropout.
solver:	str, type of 'weak' learner; currently in ('ridge', 'lasso')
activation:	str, activation function: currently 'relu', 'relu6', 'sigmoid', 'tanh'

Value

An object of class LSBoostRegressor

Examples

```
## Not run:
library(datasets)

X <- as.matrix(datasets::mtcars[, -1])
y <- as.integer(datasets::mtcars[, 1])

n <- dim(X)[1]
p <- dim(X)[2]
set.seed(21341)
train_index <- sample(x = 1:n, size = floor(0.8*n), replace = TRUE)
test_index <- -train_index
X_train <- as.matrix(X[train_index, ])
y_train <- as.double(y[train_index])
X_test <- as.matrix(X[test_index, ])
y_test <- as.double(y[test_index])

obj <- mlsauce::LSBoostRegressor()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))
## End(Not run)
```

RidgeRegressor *Ridge regressor*

Description

Ridge regressor

Usage

```
RidgeRegressor(reg_lambda = 0.1)
```

Arguments

reg_lambda	L2 regularization parameter

Value

An object of class Ridge

Examples

```
## Not run:
library(datasets)

X <- as.matrix(datasets::mtcars[, -1])
y <- as.integer(datasets::mtcars[, 1])

n <- dim(X)[1]
p <- dim(X)[2]
set.seed(21341)
train_index <- sample(x = 1:n, size = floor(0.8*n), replace = TRUE)
test_index <- -train_index
X_train <- as.matrix(X[train_index, ])
y_train <- as.double(y[train_index])
X_test <- as.matrix(X[test_index, ])
y_test <- as.double(y[test_index])


obj <- mlsauce::RidgeRegressor()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))
## End(Not run)
```

StumpClassifier *Stump classifier*

Description

Stump classifier

Usage

```
StumpClassifier(bins = "auto")
```

Arguments

bins:	int, number of histogram bins.

Value

An object of class StumpClassifier

Examples

```
## Not run:
library(datasets)

X <- as.matrix(iris[, 1:4])
y <- as.integer(iris[, 5]) - 1L

n <- dim(X)[1]
p <- dim(X)[2]
set.seed(21341)
train_index <- sample(x = 1:n, size = floor(0.8*n), replace = TRUE)
test_index <- -train_index
X_train <- as.matrix(iris[train_index, 1:4])
y_train <- as.integer(iris[train_index, 5]) - 1L
X_test <- as.matrix(iris[test_index, 1:4])
y_test <- as.integer(iris[test_index, 5]) - 1L


obj <- mlsauce::StumpClassifier()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))
## End(Not run)
```

Index