

# Package ‘mlsauce’

July 7, 2020

**Title** Miscellaneous Statistical/Machine Learning stuff

**Version** 0.4.0

**Author** T. Moudiki

**Maintainer** T. Moudiki <thierry.moudiki@pm.me>

**Description** Miscellaneous Statistical/Machine Learning stuff.

**License** BSD\_3\_clause Clear + file LICENSE

**Imports** reticulate, R6, Rcpp

**Suggests** reticulate, R6, Rcpp

**Collate** 'zzz.R' 'adaopt.R' 'lsboost.R' 'stump.R'

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.1

**NeedsCompilation** no

## R topics documented:

AdaOpt . . . . .	2
LSBoostClassifier . . . . .	3
LSBoostRegressor . . . . .	5
StumpClassifier . . . . .	6
<b>Index</b>	<b>8</b>

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**

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

```
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))
```

---

LSBoostClassifier	<i>LSBoost classifier</i>
-------------------	---------------------------

---

**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
)

```

### 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.

### 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(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::LSBoostClassifier()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))

```

---

LSBoostRegressor	<i>LSBoost Regressor</i>
------------------	--------------------------

---

## 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
)

```

## 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.

### Value

An object of class LSBoostRegressor

### Examples

```

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.integer(y[train_index, ])
X_test <- as.matrix(X[test_index, ])
y_test <- as.integer(y[test_index])

obj <- mlsauce::LSBoostRegressor()

print(obj$get_params())

obj$fit(X_train, y_train)

print(obj$score(X_test, y_test))

```

---

StumpClassifier

*Stump classifier*

---

### Description

Stump classifier

### Usage

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

**Arguments**

bins: int, number of histogram bins.

**Value**

An object of class StumpClassifier

**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(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))
```

# Index

AdaOpt, [2](#)

LSBoostClassifier, [3](#)

LSBoostRegressor, [5](#)

StumpClassifier, [6](#)