Package 'MlBayesOpt'

October 23, 2017

Type Package

Title Hyper Parameter Tuning for Machine Learning, Using Bayesian Optimization

Version 0.3.3

Maintainer Yuya Matsumura <mattu.yuya@gmail.com>

Description Hyper parameter tuning using Bayesian optimization (Shahriari et al. <doi:10.1109/JPROC.2015.2494218>) for support vector machine, random forest, and extreme gradient boosting (Chen & Guestrin (2016) <doi:10.1145/2939672.2939785>).
Unlike already existing packages (e.g. 'mlr', 'rBayesianOptimization', or 'xgboost'), there is no need to change in accordance with the package or method of machine learning.
You just prepare a data frame with feature vectors and the label column that has any class ('character', 'factor', 'integer').
Moreover, to write a optimization function, you have only to specify the data and the column name of the label to classify.

Depends R (>= 3.1.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports xgboost(>= 0.6-4), Matrix, rBayesianOptimization(>= 1.1.0), e1071(>= 1.6-8), ranger(>= 0.8.0), data.table(>= 1.9.6), foreach, rlang(>= 0.1.2), dplyr(>= 0.7.0)

Suggests MASS, testthat, knitr, rmarkdown

RoxygenNote 6.0.1

URL https://github.com/ymattu/MlBayesOpt

BugReports https://github.com/ymattu/MlBayesOpt/issues

VignetteBuilder knitr

NeedsCompilation no

Author Yuya Matsumura [aut, cre]

Repository CRAN

Date/Publication 2017-10-23 13:14:12 UTC

R topics documented:

`fashion` *Reproduced from fashion-mnist data*

Description

Reproduced from fashion-mnist data

Usage

```
fashion
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 2000 rows and 785 columns.

Source

https://github.com/ymattu/fashion-mnist-csv

`fashion_test` *Reproduced from fashion-mnist data*

Description

Reproduced from fashion-mnist data

Usage

```
fashion_test
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1000 rows and 785 columns.

Source

https://github.com/ymattu/fashion-mnist-csv

`fashion_train` *Reproduced from fashion-mnist data*

Description

Reproduced from fashion-mnist data

Usage

```
fashion_train
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1000 rows and 785 columns.

Source

https://github.com/ymattu/fashion-mnist-csv

`iris_test` *Even-numbered rows of iris data*

Description

Even-numbered rows of iris data

Usage

```
iris_test
```

Format

An object of class `data.frame` with 75 rows and 5 columns.

iris_train *Odd-numbered rows of iris data*

Description

Odd-numbered rows of iris data

Usage

```
iris_train
```

Format

An object of class `data.frame` with 75 rows and 5 columns.

rf_opt *Bayesian Optimization for Random Forest*

Description

This function estimates parameters for Random Forest based on bayesian optimization.

Usage

```
rf_opt(train_data, train_label, test_data, test_label, num_tree = 500L,
  mtry_range = c(1L, ncol(train_data) - 1), min_node_size_range = c(1L,
  as.integer(sqrt(nrow(train_data)))), init_points = 4, n_iter = 10,
  acq = "ei", kappa = 2.576, eps = 0, optkernel = list(type =
  "exponential", power = 2))
```

Arguments

`train_data`	A data frame for training of Random Forest
`train_label`	The column of class to classify in the training data
`test_data`	A data frame for training of xgboost
`test_label`	The column of class to classify in the test data
`num_tree`	The range of the number of trees for forest. Defaults to 500 (no optimization).
`mtry_range`	Value of mtry used. Defaults from 1 to number of features.
`min_node_size_range`	The range of minimum node sizes to best tested. Default min is 1 and max is sqrt(nrow(train_data)).
`init_points`	Number of randomly chosen points to sample the target function before Bayesian Optimization fitting the Gaussian Process.

n_iter	Total number of times the Bayesian Optimization is to repeated.
acq	Acquisition function type to be used. Can be "ucb", "ei" or "poi". • ucb GP Upper Confidence Bound • ei Expected Improvement • poi Probability of Improvement
kappa	tunable parameter kappa of GP Upper Confidence Bound, to balance exploitation against exploration, increasing kappa will make the optimized hyperparameters pursuing exploration.
eps	tunable parameter epsilon of Expected Improvement and Probability of Improvement, to balance exploitation against exploration, increasing epsilon will make the optimized hyperparameters are more spread out across the whole range.
optkernel	Kernel (aka correlation function) for the underlying Gaussian Process. This parameter should be a list that specifies the type of correlation function along with the smoothness parameter. Popular choices are square exponential (default) or matern 5/2

Value

The test accuracy and a list of Bayesian Optimization result is returned:

- Best_Par a named vector of the best hyperparameter set found
- Best_Value the value of metrics achieved by the best hyperparameter set
- History a data.table of the bayesian optimization history
- Pred a data.table with validation/cross-validation prediction for each round of bayesian optimization history

Examples

```
library(MlBayesOpt)

set.seed(71)
res0 <- rf_opt(train_data = iris_train,
               train_label = Species,
               test_data = iris_test,
               test_label = Species,
               mtry_range = c(1L, ncol(iris_train) - 1),
               num_tree = 10L,
               init_points = 10,
               n_iter = 1)
```

svm_cv_opt *Bayesian Optimization for SVM*

Description

This function estimates parameters for SVM(Gaussian Kernel) based on bayesian optimization

Usage

```
svm_cv_opt(data, label, gamma_range = c(10^(-3), 10^1),
  cost_range = c(10^(-2), 10^2), svm_kernel = "radial",
  degree_range = c(3L, 10L), coef0_range = c(10^(-1), 10^1), n_folds,
  init_points = 4, n_iter = 10, acq = "ei", kappa = 2.576, eps = 0,
  optkernel = list(type = "exponential", power = 2))
```

Arguments

data	data
label	label for classification
gamma_range	The range of gamma. Default is c(10 ^ (-3), 10 ^ 1)
cost_range	The range of C(Cost). Default is c(10 ^ (-2), 10 ^ 2)
svm_kernel	Kernel used in SVM. You might consider changing some of the following parameters, depending on the kernel type. • **linear:** $u'v$ • **polynomial:** $(\gamma u'v + coef0)^{degree}$ • **radial basis:** $exp(-\gamma\|u-v\|^2)$ • **sigmoid:** $tanh(\gamma u'v + coef0)$
degree_range	Parameter needed for kernel of type polynomial. Default is c(3L, 10L)
coef0_range	Parameter needed for kernels of type `polynomial` and `sigmoid`. Default is c(10 ^ (-1), 10 ^ 1)
n_folds	if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification and the Mean Squared Error for regression
init_points	Number of randomly chosen points to sample the target function before Bayesian Optimization fitting the Gaussian Process.
n_iter	Total number of times the Bayesian Optimization is to repeated.
acq	Acquisition function type to be used. Can be "ucb", "ei" or "poi". • `ucb` GP Upper Confidence Bound • `ei` Expected Improvement • `poi` Probability of Improvement
kappa	tunable parameter kappa of GP Upper Confidence Bound, to balance exploitation against exploration, increasing kappa will make the optimized hyperparameters pursuing exploration.

eps	tunable parameter epsilon of Expected Improvement and Probability of Improvement, to balance exploitation against exploration, increasing epsilon will make the optimized hyperparameters are more spread out across the whole range.
optkernel	Kernel (aka correlation function) for the underlying Gaussian Process. This parameter should be a list that specifies the type of correlation function along with the smoothness parameter. Popular choices are square exponential (default) or matern 5/2

Value

The test accuracy and a list of Bayesian Optimization result is returned:

- `Best_Par` a named vector of the best hyperparameter set found
- `Best_Value` the value of metrics achieved by the best hyperparameter set
- `History` a `data.table` of the bayesian optimization history
- `Pred` a `data.table` with validation/cross-validation prediction for each round of bayesian optimization history

Examples

```
library(MlBayesOpt)

set.seed(71)
res0 <- svm_cv_opt(data = iris,
                   label = Species,
                   n_folds = 3,
                   init_points = 10,
                   n_iter = 1)
```

`svm_opt` *Bayesian Optimization for SVM*

Description

This function estimates parameters for SVM(Gaussian Kernel) based on bayesian optimization

Usage

```
svm_opt(train_data, train_label, test_data, test_label,
  gamma_range = c(10^(-3), 10^1), cost_range = c(10^(-2), 10^2),
  svm_kernel = "radial", degree_range = c(3L, 10L),
  coef0_range = c(10^(-1), 10^1), init_points = 4, n_iter = 10,
  acq = "ei", kappa = 2.576, eps = 0, optkernel = list(type =
  "exponential", power = 2))
```

Arguments

train_data	A data frame for training of SVM
train_label	The column of class to classify in the training data
test_data	A data frame for training of SVM
test_label	The column of class to classify in the test data
gamma_range	The range of gamma. Default is c(10 ^ (-3), 10 ^ 1)
cost_range	The range of C(Cost). Default is c(10 ^ (-2), 10 ^ 2)
svm_kernel	Kernel used in SVM. You might consider changing some of the following parameters, depending on the kernel type. • **linear:** $u'v$ • **polynomial:** $(\gamma u'v + coef0)^{degree}$ • **radial basis:** $exp(-\gamma\|u - v\|^2)$ • **sigmoid:** $tanh(\gamma u'v + coef0)$
degree_range	Parameter needed for kernel of type polynomial. Default is c(3L, 10L)
coef0_range	parameter needed for kernels of type `polynomial` and `sigmoid`. Default is c(10 ^ (-1), 10 ^ 1)
init_points	Number of randomly chosen points to sample the target function before Bayesian Optimization fitting the Gaussian Process.
n_iter	Total number of times the Bayesian Optimization is to repeated.
acq	Acquisition function type to be used. Can be "ucb", "ei" or "poi". • `ucb` GP Upper Confidence Bound • `ei` Expected Improvement • `poi` Probability of Improvement
kappa	tunable parameter kappa of GP Upper Confidence Bound, to balance exploitation against exploration, increasing kappa will make the optimized hyperparameters pursuing exploration.
eps	tunable parameter epsilon of Expected Improvement and Probability of Improvement, to balance exploitation against exploration, increasing epsilon will make the optimized hyperparameters are more spread out across the whole range.
optkernel	Kernel (aka correlation function) for the underlying Gaussian Process. This parameter should be a list that specifies the type of correlation function along with the smoothness parameter. Popular choices are square exponential (default) or matern 5/2

Value

The test accuracy and a list of Bayesian Optimization result is returned:

- `Best_Par` a named vector of the best hyperparameter set found
- `Best_Value` the value of metrics achieved by the best hyperparameter set
- `History` a `data.table` of the bayesian optimization history
- `Pred` a `data.table` with validation/cross-validation prediction for each round of bayesian optimization history

Examples

```
library(MlBayesOpt)

set.seed(71)
res0 <- svm_opt(train_data = iris_train,
                train_label = Species,
                test_data = iris_test,
                test_label = Species,
                svm_kernel = "polynomial",
                init_points = 10,
                n_iter = 1)
```

---

xgb_cv_opt *Bayesian Optimization for XGboost(Cross Validation)*

Description

Bayesian Optimization for XGboost (Cross Validation)

Usage

```
xgb_cv_opt(data, label, objectfun, evalmetric, n_folds, eta_range = c(0.1,
  1L), max_depth_range = c(4L, 6L), nrounds_range = c(70, 160L),
  subsample_range = c(0.1, 1L), bytree_range = c(0.4, 1L),
  init_points = 4, n_iter = 10, acq = "ei", kappa = 2.576, eps = 0,
  optkernel = list(type = "exponential", power = 2), classes = NULL,
  seed = 0)
```

Arguments

`data`	data
`label`	label for classification
`objectfun`	Specify the learning task and the corresponding learning objective • `reg:linear` linear regression (Default). • `reg:logistic` logistic regression. • `binary:logistic` logistic regression for binary classification. Output probability. • `binary:logitraw` logistic regression for binary classification, output score before logistic transformation. • `multi:softmax` set xgboost to do multiclass classification using the softmax objective. Class is represented by a number and should be from 0 to `num_class - 1`. • `multi:softprob` same as softmax, but prediction outputs a vector of ndata * nclass elements, which can be further reshaped to ndata, nclass matrix. The result contains predicted probabilities of each data point belonging to each class.

- `rank:pairwise` set xgboost to do ranking task by minimizing the pairwise loss.

`evalmetric` evaluation metrics for validation data. Users can pass a self-defined function to it. Default: metric will be assigned according to objective(rmse for regression, and error for classification, mean average precision for ranking).

- `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

`n_folds` K for cross Validation

`eta_range` The range of eta(default is c(0.1, 1L))

`max_depth_range` The range of max_depth(default is c(4L, 6L))

`nrounds_range` The range of nrounds(default is c(70, 160L))

`subsample_range` The range of subsample rate(default is c(0.1, 1L))

`bytree_range` The range of colsample_bytree rate(default is c(0.4, 1L)

`init_points` Number of randomly chosen points to sample the target function before Bayesian Optimization fitting the Gaussian Process.

`n_iter` Total number of times the Bayesian Optimization is to repeated.

`acq` Acquisition function type to be used. Can be "ucb", "ei" or "poi". #'

- `ucb` GP Upper Confidence Bound
- `ei` Expected Improvement
- `poi` Probability of Improvement

`kappa` kappa tunable parameter kappa of GP Upper Confidence Bound, to balance exploitation against exploration, increasing kappa will make the optimized hyperparameters pursuing exploration.

`eps` tunable parameter epsilon of Expected Improvement and Probability of Improvement, to balance exploitation against exploration, increasing epsilon will make the optimized hyperparameters are more spread out across the whole range.

`optkernel` Kernel (aka correlation function) for the underlying Gaussian Process. This parameter should be a list that specifies the type of correlation function along with the smoothness parameter. Popular choices are square exponential (default) or matern 5/2

`classes` set the number of classes. To use only with multiclass objectives.

`seed` set seed.(default is 0)

Value

The score you specified in the evalmetric option and a list of Bayesian Optimization result is returned:

- `Best_Par` a named vector of the best hyperparameter set found
- `Best_Value` the value of metrics achieved by the best hyperparameter set
- `History` a `data.table` of the bayesian optimization history
- `Pred` a `data.table` with validation/cross-validation prediction for each round of bayesian optimization history

Examples

```
library(MlBayesOpt)

set.seed(71)
res0 <- xgb_cv_opt(data = iris,
                   label = Species,
                   objectfun = "multi:softmax",
                   evalmetric = "mlogloss",
                   n_folds = 3,
                   classes = 3,
                   init_points = 2,
                   n_iter = 1)
```

---

xgb_opt *Bayesian Optimization for XGboost*

Description

This function estimates parameters for xgboost based on bayesian optimization.

Usage

```
xgb_opt(train_data, train_label, test_data, test_label, objectfun, evalmetric,
  eta_range = c(0.1, 1L), max_depth_range = c(4L, 6L),
  nrounds_range = c(70, 160L), subsample_range = c(0.1, 1L),
  bytree_range = c(0.4, 1L), init_points = 4, n_iter = 10, acq = "ei",
  kappa = 2.576, eps = 0, optkernel = list(type = "exponential", power =
  2), classes = NULL)
```

Arguments

`train_data`	A data frame for training of xgboost
`train_label`	The column of class to classify in the training data
`test_data`	A data frame for training of xgboost
`test_label`	The column of class to classify in the test data
`objectfun`	Specify the learning task and the corresponding learning objective • `reg:linear` linear regression (Default). • `reg:logistic` logistic regression.

- `binary:logistic` logistic regression for binary classification. Output probability.
- `binary:logitraw` logistic regression for binary classification, output score before logistic transformation.
- `multi:softmax` set xgboost to do multiclass classification using the softmax objective. Class is represented by a number and should be from 0 to `num_class - 1`.
- `multi:softprob` same as softmax, but prediction outputs a vector of ndata * nclass elements, which can be further reshaped to ndata, nclass matrix. The result contains predicted probabilities of each data point belonging to each class.
- `rank:pairwise` set xgboost to do ranking task by minimizing the pairwise loss.

`evalmetric` evaluation metrics for validation data. Users can pass a self-defined function to it. Default: metric will be assigned according to objective(rmse for regression, and error for classification, mean average precision for ranking).

- `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

`eta_range` The range of eta

`max_depth_range`
The range of max_depth

`nrounds_range` The range of nrounds

`subsample_range`
The range of subsample rate

`bytree_range` The range of colsample_bytree rate

`init_points` Number of randomly chosen points to sample the target function before Bayesian Optimization fitting the Gaussian Process.

`n_iter` Total number of times the Bayesian Optimization is to repeated.

`acq` Acquisition function type to be used. Can be "ucb", "ei" or "poi".

- `ucb` GP Upper Confidence Bound
- `ei` Expected Improvement
- `poi` Probability of Improvement

`kappa` tunable parameter kappa of GP Upper Confidence Bound, to balance exploitation against exploration, increasing kappa will make the optimized hyperparameters pursuing exploration.

`eps` tunable parameter epsilon of Expected Improvement and Probability of Improvement, to balance exploitation against exploration, increasing epsilon will make the optimized hyperparameters are more spread out across the whole range.

optkernel	Kernel (aka correlation function) for the underlying Gaussian Process. This parameter should be a list that specifies the type of correlation function along with the smoothness parameter. Popular choices are square exponential (default) or matern 5/2
classes	set the number of classes. To use only with multiclass objectives.

Value

The test accuracy and a list of Bayesian Optimization result is returned:

- `Best_Par` a named vector of the best hyperparameter set found
- `Best_Value` the value of metrics achieved by the best hyperparameter set
- `History` a `data.table` of the bayesian optimization history
- `Pred` a `data.table` with validation/cross-validation prediction for each round of bayesian optimization history

Examples

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

set.seed(71)
res0 <- xgb_opt(train_data = fashion_train,
                train_label = y,
                test_data = fashion_test,
                test_label = y,
                objectfun = "multi:softmax",
                evalmetric = "merror",
                classes = 10,
                init_points = 3,
                n_iter = 1)

## End(Not run)
```

Index