Package 'rBayesianOptimization'

September 14, 2016

Type Package

Title Bayesian Optimization of Hyperparameters

Version 1.1.0

Description A Pure R implementation of Bayesian Global Optimization with Gaussian Processes.

URL http://github.com/yanyachen/rBayesianOptimization

BugReports http://github.com/yanyachen/rBayesianOptimization/issues

Depends R (>= 2.10)

Imports stats, utils, data.table (>= 1.9.6), magrittr, foreach, GPfit

Suggests xgboost

License GPL-2

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

Author Yachen Yan [aut, cre]

Maintainer Yachen Yan <yanyachen21@gmail.com>

Repository CRAN

Date/Publication 2016-09-14 21:44:11

R topics documented:

`BayesianOptimization` *Bayesian Optimization*

Description

Bayesian Optimization of Hyperparameters.

Usage

```
BayesianOptimization(FUN, bounds, init_grid_dt = NULL, init_points = 0,
  n_iter, acq = "ucb", kappa = 2.576, eps = 0, kernel = list(type =
  "exponential", power = 2), verbose = TRUE, ...)
```

Arguments

`FUN`	The function to be maximized. This Function should return a named list with 2 components. The first component "Score" should be the metrics to be maximized, and the second component "Pred" should be the validation/cross-validation prediction for ensembling/stacking.
`bounds`	A named list of lower and upper bounds for each hyperparameter. The names of the list should be identical to the arguments of FUN. All the sample points in init_grid_dt should be in the range of bounds. Please use "L" suffix to indicate integer hyperparameter.
`init_grid_dt`	User specified points to sample the target function, should be a `data.frame` or `data.table` with identical column names as bounds. User can add one "Value" column at the end, if target function is pre-sampled.
`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.
`kernel`	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
`verbose`	Whether or not to print progress.
`...`	Other arguments passed on to `GP_fit`.

Value

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

References

Jasper Snoek, Hugo Larochelle, Ryan P. Adams (2012) *Practical Bayesian Optimization of Machine Learning Algorithms*

Examples

```
# Example 1: Optimization
## Set Pred = 0, as placeholder
Test_Fun <- function(x) {
  list(Score = exp(-(x - 2)^2) + exp(-(x - 6)^2/10) + 1/ (x^2 + 1),
       Pred = 0)
}
## Set larger init_points and n_iter for better optimization result
OPT_Res <- BayesianOptimization(Test_Fun,
                                bounds = list(x = c(1, 3)),
                                init_points = 2, n_iter = 1,
                                acq = "ucb", kappa = 2.576, eps = 0.0,
                                verbose = TRUE)
## Not run:
# Example 2: Parameter Tuning
library(xgboost)
data(agaricus.train, package = "xgboost")
dtrain <- xgb.DMatrix(agaricus.train$data,
                      label = agaricus.train$label)
cv_folds <- KFold(agaricus.train$label, nfolds = 5,
                  stratified = TRUE, seed = 0)
xgb_cv_bayes <- function(max.depth, min_child_weight, subsample) {
  cv <- xgb.cv(params = list(booster = "gbtree", eta = 0.01,
                             max_depth = max.depth,
                             min_child_weight = min_child_weight,
                             subsample = subsample, colsample_bytree = 0.3,
                             lambda = 1, alpha = 0,
                             objective = "binary:logistic",
                             eval_metric = "auc"),
               data = dtrain, nround = 100,
               folds = cv_folds, prediction = TRUE, showsd = TRUE,
               early.stop.round = 5, maximize = TRUE, verbose = 0)
  list(Score = cv$dt[, max(test.auc.mean)],
       Pred = cv$pred)
}
OPT_Res <- BayesianOptimization(xgb_cv_bayes,
```

```
                                bounds = list(max.depth = c(2L, 6L),
                                              min_child_weight = c(1L, 10L),
                                              subsample = c(0.5, 0.8)),
                                init_grid_dt = NULL, init_points = 10, n_iter = 20,
                                acq = "ucb", kappa = 2.576, eps = 0.0,
                                verbose = TRUE)

## End(Not run)
```

`KFold` *K-Folds cross validation index generator*

Description

Generates a list of indices for K-Folds Cross-Validation.

Usage

```
KFold(target, nfolds = 10, stratified = FALSE, seed = 0)
```

Arguments

`target`	Samples to split in K folds.
`nfolds`	Number of folds.
`stratified`	whether to apply Stratified KFold.
`seed`	random seed to be used.

Value

a list of indices for K-Folds Cross-Validation

`rBayesianOptimization` *rBayesianOptimization: Bayesian Optimization of Hyperparameters*

Description

A Pure R implementation of bayesian global optimization with gaussian processes.

Index