Package 'ParBayesianOptimization'

July 31, 2019

Title Parallel Bayesian Optimization of Hyperparameters

Version 0.1.2

Description Fast, flexible framework for implementing Bayesian optimization of model hyperparameters according to the methods described in Snoek et al. <arXiv:1206.2944>. The package allows the user to run scoring function in parallel, save intermediary results, and tweak other aspects of the process to fully utilize the computing resources available to the user.

URL `https://github.com/AnotherSamWilson/ParBayesianOptimization`

BugReports `https://github.com/AnotherSamWilson/ParBayesianOptimization/issues`

Depends R (>= 3.4)

Imports data.table (>= 1.11.8), GauPro, stats, foreach, dbscan, lhs

Suggests knitr,rmarkdown,xgboost,doParallel,ggplot2

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

Maintainer Samuel Wilson `<samwilson303@gmail.com>`

NeedsCompilation no

Author Samuel Wilson [aut, cre]

Repository CRAN

Date/Publication 2019-07-31 12:40:10 UTC

R topics documented:

BayesianOptimization *Bayesian Optimization*

Description

Flexible Bayesian optimization of model hyperparameters.

Usage

```
BayesianOptimization(FUN, bounds, saveIntermediate = NULL,
  leftOff = NULL, parallel = FALSE, packages = NULL, export = NULL,
  initialize = TRUE, initGrid = NULL, initPoints = 0, bulkNew = 1,
  nIters = 0, kern = "Matern52", beta = 0, acq = "ucb",
  stopImpatient = list(newAcq = "ucb", rounds = Inf), kappa = 2.576,
  eps = 0, gsPoints = 100, convThresh = 1e+07,
  minClusterUtility = NULL, noiseAdd = 0.25, verbose = 1)
```

Arguments

FUN	the function to be maximized. This function should return a named list with at least 1 component. The first component must be named Score and should contain the metric to be maximized. You may return other named scalar elements that you wish to include in the final summary table.
bounds	named list of lower and upper bounds for each hyperparameter. The names of the list should be arguments passed to FUN. Use "L" suffix to indicate integer hyperparameters.
saveIntermediate	character filepath (including file name) that specifies the location to save intermediary results. This will save a data.table as an RDS that can be specified as the leftOff parameter.
leftOff	data.table containing parameter-Score pairs. If supplied, the process will rbind this table to the parameter-Score pairs obtained through initialization. This table should be obtained from the file saved by saveIntermediate.
parallel	should the process run in parallel? If TRUE, several criteria must be met: • A parallel backend must be registered • FUN must be executable using only packages specified in packages (and base packages) • FUN must be executable using only the the objects specified in export • The function must be thread safe.
packages	character vector of the packages needed to run FUN.
export	character vector of object names needed to evaluate FUN.
initialize	should the process initialize a parameter-Score pair set? If FALSE, leftOff must be provided.

initGrid — user specified points to sample the scoring function, should be a data.frame or data.table with identical column names as bounds.

initPoints — number of randomly chosen points to sample the scoring function before Bayesian Optimization fitting the Gaussian Process.

bulkNew — integer that specifies the number of parameter combinations to try between each Gaussian process fit.

nIters — total number of parameter sets to be sampled, including initial set.

kern — a character that gets mapped to one of GauPro's GauPro_kernel_beta S6 classes. Determines the covariance function used in the gaussian process. Can be one of:

- "Gaussian"
- "Exponential"
- "Matern52"
- "Matern32"

beta — the kernel lengthscale parameter log10(theta). Passed to GauPro_kernel_beta specified in kern.

acq — acquisition function type to be used. Can be "ucb", "ei", "eips" or "poi".

- ucb Upper Confidence Bound
- ei Expected Improvement
- eips Expected Improvement Per Second
- poi Probability of Improvement

stopImpatient — a list containing rounds and newAcq, if acq = "eips" you can switch the acquisition function to newAcq after rounds parameter-score pairs are found.

kappa — tunable parameter kappa of the upper confidence bound. Tunes exploitation/exploration. Increasing kappa will increase the importance that variance (unexplored space) has, therefore incentivising exploration.

eps — tunable parameter epsilon of ei, eips and poi. Tunes exploitation/exploration. Increasing eps will make the "improvement" threshold higher.

gsPoints — integer that specifies how many initial points to try when searching for the optimal parameter set. Increase this for a higher chance to find global optimum, at the expense of more time.

convThresh — convergence threshold passed to factr when the optim function (L-BFGS-B) is called. Lower values will take longer to converge, but may be more accurate.

minClusterUtility — number 0-1. Represents the minimum percentage of the optimal utility required for a less optimal local maximum to be included as a candidate parameter set in the next scoring function. If NULL, only the global optimum will be used as a candidate parameter set.

noiseAdd — if bulkNew > 1, specifies how much noise to add to the optimal candidate parameter set to obtain the other bulkNew-1 candidate parameter sets. New random draws are pulled from a shape(4,4) beta distribution centered at the optimal candidate parameter set with a range equal to noiseAdd*(Upper Bound - Lower Bound)

verbose	Whether or not to print progress. If 0, nothing will be printed. If 1, progress will be printed. If 2, progress and information about new parameter-score pairs will be printed.

Value

A list containing details about the process:

GPs	The last Gaussian Process run on the parameter-score pairs
GPe	If acq = "eips", this contains the last Gaussian Process run on the parameter-elapsed time pairs
acqMaximums	The optimal parameters according to each gaussian process
ScoreDT	A list of all parameter-score pairs, as well as extra columns from FUN
BestPars	The best parameter set at each iteration

References

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

Examples

```
# Example 1 - Optimization of a Linear Function
scoringFunction <- function(x) {
  a <- exp(-(2-x)^2)*1.5
  b <- exp(-(4-x)^2)*2
  c <- exp(-(6-x)^2)*1
  return(list(Score = a+b+c))
}

bounds <- list(x = c(0,8))

Results <- BayesianOptimization(
    FUN = scoringFunction
  , bounds = bounds
  , initPoints = 5
  , nIters = 8
  , gsPoints = 10
)

## Not run:
# Example 2 - Hyperparameter Tuning in xgboost
library("xgboost")

data(agaricus.train, package = "xgboost")

Folds <- list( Fold1 = as.integer(seq(1,nrow(agaricus.train$data),by = 3))
             , Fold2 = as.integer(seq(2,nrow(agaricus.train$data),by = 3))
             , Fold3 = as.integer(seq(3,nrow(agaricus.train$data),by = 3)))

scoringFunction <- function(max_depth, min_child_weight, subsample) {
```

```
  dtrain <- xgb.DMatrix(agaricus.train$data,label = agaricus.train$label)

  Pars <- list( booster = "gbtree"
              , eta = 0.01
              , max_depth = max_depth
              , min_child_weight = min_child_weight
              , subsample = subsample
              , objective = "binary:logistic"
              , eval_metric = "auc")

  xgbcv <- xgb.cv( params = Pars
                 , data = dtrain
                 , nround = 100
                 , folds = Folds
                 , prediction = TRUE
                 , showsd = TRUE
                 , early_stopping_rounds = 5
                 , maximize = TRUE
                 , verbose = 0)

  return(list( Score = max(xgbcv$evaluation_log$test_auc_mean)
             , nrounds = xgbcv$best_iteration
  )
  )
}

bounds <- list(max_depth = c(2L, 10L)
             , min_child_weight = c(1, 100)
             , subsample = c(0.25, 1))

kern <- "Matern52"

acq <- "ei"

ScoreResult <- BayesianOptimization(
    FUN = scoringFunction
  , bounds = bounds
  , initPoints = 10
  , bulkNew = 1
  , nIters = 12
  , kern = kern
  , acq = acq
  , kappa = 2.576
  , verbose = 1
  , parallel = FALSE
  , gsPoints = 50)

## End(Not run)
```