Package 'randomsearch'

January 17, 2019

Title Random Search for Expensive Functions

Description Simple Random Search function for the 'smoof' and 'ParamHelpers' ecosystem with termination criteria and parallelization.

License BSD_2_clause + file LICENSE

URL https://github.com/jakob-r/randomsearch

BugReports https://github.com/jakob-r/randomsearch/issues

Depends ParamHelpers (>= 1.10), smoof (>= 1.5.1)

Imports checkmate (>= 1.8.2), parallelMap (>= 1.3), fs

Suggests emoa, rmarkdown, testthat, knitr, roxygen2

LazyData yes

Encoding UTF-8

ByteCompile yes

Version 0.2.0

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Jakob Richter [aut, cre] (<https://orcid.org/0000-0003-4481-5554>)

Maintainer Jakob Richter <code@jakob-r.de>

Repository CRAN

Date/Publication 2019-01-17 12:50:03 UTC

R topics documented:

`randomsearch` *Optimizes a function with random search.*

Description

This function conducts a random search on the given function with the support of parallelization and multiple termination criteria.

Usage

```
randomsearch(fun, ...)

## S3 method for class 'smoof_function'
randomsearch(fun, design = NULL,
  max.evals = 20, max.execbudget = NULL, target.fun.value = NULL,
  design.y.cols = NULL, par.dir = NULL, par.jobs = NULL, ...)

## S3 method for class 'function'
randomsearch(fun, minimize = TRUE, lower, upper,
  design = NULL, max.evals = 20, max.execbudget = NULL,
  target.fun.value = NULL, design.y.cols = NULL, par.dir = NULL,
  par.jobs = NULL, ...)
```

Arguments

`fun` [`smoof_function|function`]
Fitness function to optimize. Can be either a `smoof_function` or a normal function that takes the numeric vector over that should be optimized as the first argument. For one dimensional target functions you can obtain a `smoof_function` by using `makeSingleObjectiveFunction`. For multi dimensional functions use `makeMultiObjectiveFunction`. It is possible to return even more information which will be stored in the optimization path. To achieve this, simply append the attribute "extras" to the return value of the target function. This has to be a named list of scalar values. Each of these values will be stored additionally in the optimization path.

`...` [any]
Additional arguments that will be passed to each call of `fun`.

`design` [`data.frame`]
Initial design as data frame. If the y-values are not already present in design, randomsearch will evaluate the points. If the parameters have corresponding trafo functions, the design must not be transformed before it is passed! Functions to generate designs are available in `ParamHelpers`: `generateDesign`, `generateGridDesign`, `generateRandomDesign`. Default is `NULL`, which means no initial design.

`max.evals` [`integer(1)`]
Maximum number of evaulations of the objective functions. Includes the initial design.

`max.execbudget` [integer(1)]
Exceution time budget in seconds.

`target.fun.value`
[numeric(1)]
Traget function value.

`design.y.cols` [characer()]
The name of the column containing the function outcomes. One for single-crit optimization. Multiple for multi-crit optimization.

`par.dir` [character(1)]
Location to store parallel communication files. Defaults to `tmpfile()` which might not be suitable for parallelization methods that work on multiple machines. Those need a shared directory.

`par.jobs` [integer(1)]
How many parallel jobs do you want to run to evaluate the random search? Default is `NULL` which means 1 if no `parallelStart*` function is called. Otherwise it will detect the number through `parallelGetOptions`.

`minimize` [logical()]
Wheter the function should be minimized or maximized. If this is a vector we will assume it is a multi-objective optimization. Has to have the same length as the output of the objective function `fun`.

`lower` [numeric()]
Lower bounds on the variables.

`upper` [numeric()]
Upper bounds on the variables.

Value

`OptPath`

Methods (by class)

- `smoof_function`: optimize smoof function
- `function`: optimize generic function

Examples

```
obj.fun = makeSingleObjectiveFunction(
 fn = function(x) x[1]^2 + sin(x[2]),
 par.set = makeNumericParamSet(id = "x", lower = -1, upper = 1, len = 2)
)
res = randomsearch(obj.fun, max.evals = 10)
summary(res)
```

summary.RandomsearchResult *Generate result summary*

Description

Generate results from randomsearch run. For normal single objective runs the best result is returned. For mulit-objective runs the pareto front is returned.

Usage

```
## S3 method for class 'RandomsearchResult'
summary(object, ...)
```

Arguments

object	[OptPath] Optimization path.
...	[any] Currently ignored.

Index