Package 'ao'

January 4, 2022

Title Alternating Optimization

Version 0.2.1

Date 2022-04-01

Description Alternating optimization of (high-dimensional) functions is an iterative procedure for minimizing (or maximizing) jointly over all parameters by alternately optimizing parameter subsets. For a reference, see Bezdek and Hathaway (2002) ``Some Notes on Alternating Optimization'' <doi:10.1007/3-540-45631-7_39>.

URL https://loelschlaeger.github.io/ao/

BugReports https://github.com/loelschlaeger/ao/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

Imports R.utils, optimx, ggplot2

Suggests testthat (>= 3.0.0), magrittr, knitr, rmarkdown, covr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Lennart Oelschläger [aut, cre] (<https://orcid.org/0000-0001-5421-9313>)

Maintainer Lennart Oelschläger <lennart.oelschlaeger@uni-bielefeld.de>

Repository CRAN

Date/Publication 2022-01-04 16:30:02 UTC

R topics documented:

---

ao *Alternating Optimization.*

Description

This function performs alternating optimization on the function `f`.

Usage

```
ao(
  f,
  partition,
  initial = 0,
  iterations = 10,
  tolerance = 1e-06,
  minimize = TRUE,
  progress = FALSE,
  plot = TRUE
)
```

Arguments

`f`	An object of class `ao_f`, i.e. the output of `set_f`.
`partition`	A list of vectors of parameter indices $1, ..., n$ of the function. For example, choosing `partition = list(1,2)` as in the example optimizes each parameter separately, while choosing `partition = list(1:2)` leads to joint optimization. Parameter indices can be members of multiple subsets.
`initial`	A vector of length `f$npar` of initial parameter values. Per default, the algorithm is initialized at the origin.
`iterations`	The number of iterations through all subsets.
`tolerance`	A non-negative numeric value. The function terminates prematurely if the euclidean distance between the current solution and the one from the last iteration is smaller than `tolerance`.
`minimize`	If `TRUE`, minimization, if `FALSE`, maximization.
`progress`	If `TRUE`, progress is printed.
`plot`	If `TRUE`, the parameter updates are plotted.

Details

This function depends on `optimx`.

Value

An object of class ao, which is a list of

- optimum, the optimal value,
- estimate, the parameter vector that yields the optimum,
- sequence, a data frame of the estimates in the single iterations,
- time, the total estimation time in seconds.

Examples

```
himmelblau <- function(x) (x[1]^2 + x[2] - 11)^2 + (x[1] + x[2]^2 - 7)^2
f <- set_f(f = himmelblau, npar = 2, lower = -5, upper = 5)
ao(f = f, partition = list(1, 2))
```

---

euclidean *Euclidean distance.*

Description

This function computes the euclidean distance between two numeric vectors.

Usage

```
euclidean(a, b)
```

Arguments

a	A numeric (vector).
b	A numeric (vector).

Value

A numeric.

Examples

```
euclidean(c(0, 0), c(1, 1))
```

is_number *Check for number.*

Description

This function checks if the input `x` is a (vector of) number(s), i.e. a (vector of) positive integer value(s).

Usage

```
is_number(x)
```

Arguments

`x`	A (vector of) numeric value(s).

Value

A logical vector of the same length as `x`.

Examples

```
is_number(c(0, 1, 1.5))
```

set_f *Specify function.*

Description

This function specifies the function to be optimized.

Usage

```
set_f(f, ..., npar, lower = -Inf, upper = Inf, iterlim = NULL, check = FALSE)
```

Arguments

`f`	A function to be optimized, returning a single numeric value. Its first argument should be a numeric vector of length `npar`. Additional arguments can be specified via the `...` argument. Gradient or Hessian of `f` can be specified via attributes `gradient` and `hessian` for the function value. They are used for optimization if specified.
`...`	Additional arguments to be passed to `f`.
`npar`	The number of variables of `f`.

lower	Lower bounds on the variables, which can be a single numeric value (a joint bound for all parameters) or a numeric vector of length npar (for individual bounds).
upper	Upper bounds on the variables, analogue to lower.
iterlim	The maximum number of iterations for the numerical optimizer for each subproblem. No limit per default.
check	If TRUE checks the configuration. This will take at most 20 seconds in most cases. Set to FALSE if you are confident about the configuration to save computation time.

Value

An object of class ao_f.

Examples

```
himmelblau <- function(x) (x[1]^2 + x[2] - 11)^2 + (x[1] + x[2]^2 - 7)^2
set_f(f = himmelblau, npar = 2, lower = -5, upper = 5)
```

---

timed *Interruption of long evaluations.*

Description

This function evaluates expr and interrupts the evaluation after secs seconds.

Usage

```
timed(expr, secs)
```

Arguments

expr	An R expression to evaluate.
secs	The number of seconds.

Details

This function is a wrapper for withTimeout.

Value

Either the value of expr or NULL if the evaluation time exceeded secs seconds.

Examples

```
timed(Sys.sleep(1.1), 1)
```

`try_silent` *Try an expression silently.*

Description

This function tries to execute `expr` and returns a string with the error message if the execution failed.

Usage

```
try_silent(expr)
```

Arguments

`expr`	An R expression to try.

Details

This function is a wrapper for `try`.

Value

Either the value of `expr` or in case of a failure an object of class `ao_fail`, which is the error message.

Examples

```
try_silent(log(1))
try_silent(log("1"))
```

Index