Package ‘cmaes’

October 12, 2022

Title Covariance Matrix Adapting Evolutionary Strategy

Version 1.0-12

Author Heike Trautmann <trautmann@statistik.tu-dortmund.de> and Olaf Mersmann <olafm@statistik.tu-dortmund.de> and David Arnu <david.arnu@statistik.tu-dortmund.de>

Maintainer Olaf Mersmann <olafm@statistik.tu-dortmund.de>

Description Single objective optimization using a CMA-ES.

License GPL-2

Depends R (>= 2.9.0)

Suggests RUnit

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-18 10:20:18 UTC

R topics documented:

bias_function *Create a biased test function...*

Description

Create a biased test function

Usage

```
bias_function(f, bias)
```

Arguments

f	test function
bias	bias value.

Details

Returns a new biased test function defined as

$$g(x) = f(x) + bias.$$

Value

The biased test function.

Author(s)

Olaf Mersmann <olafm@statistik.tu-dortmund.de>

cma_es *Covariance matrix adapting evolutionary strategy*

Description

Global optimization procedure using a covariance matrix adapting evolutionary strategy.

Usage

```
cma_es(par, fn, ..., lower, upper, control=list())
cmaES(...)
```

Arguments

`par`	Initial values for the parameters to be optimized over.
`fn`	A function to be minimized (or maximized), with first argument the vector of parameters over which minimization is to take place. It should return a scalar result.
`...`	Further arguments to be passed to `fn`.
`lower`	Lower bounds on the variables.
`upper`	Upper bounds on the variables.
`control`	A list of control parameters. See 'Details'.

Details

`cma_es`: Note that arguments after `...` must be matched exactly. By default this function performs minimization, but it will maximize if `control$fnscale` is negative. It can usually be used as a drop in replacement for `optim`, but do note, that no sophisticated convergence detection is included. Therefore you need to choose `maxit` appropriately.

If you set `vectorize==TRUE`, `fn` will be passed matrix arguments during optimization. The columns correspond to the `lambda` new individuals created in each iteration of the ES. In this case `fn` must return a numeric vector of `lambda` corresponding function values. This enables you to do up to `lambda` function evaluations in parallel.

The `control` argument is a list that can supply any of the following components:

`fnscale` An overall scaling to be applied to the value of `fn` during optimization. If negative, turns the problem into a maximization problem. Optimization is performed on `fn(par)/fnscale`.

`maxit` The maximum number of iterations. Defaults to $100 * D^2$, where D is the dimension of the parameter space.

`stopfitness` Stop if function value is smaller than or equal to `stopfitness`. This is the only way for the CMA-ES to "converge".

keep.best return the best overall solution and not the best solution in the last population. Defaults to true.

`sigma` Initial variance estimates. Can be a single number or a vector of length D, where D is the dimension of the parameter space.

`mu` Population size.

`lambda` Number of offspring. Must be greater than or equal to `mu`.

`weights` Recombination weights

`damps` Damping for step-size

`cs` Cumulation constant for step-size

`ccum` Cumulation constant for covariance matrix

`vectorized` Is the function `fn` vectorized?

`ccov.1` Learning rate for rank-one update

`ccov.mu` Learning rate for rank-mu update

`diag.sigma` Save current step size σ in each iteration.

`diag.eigen` Save current principle components of the covariance matrix C in each iteration.

`diag.pop` Save current population in each iteration.

`diag.value` Save function values of the current population in each iteration.

Value

`cma_es`: A list with components:

par The best set of parameters found.

value The value of `fn` corresponding to `par`.

counts A two-element integer vector giving the number of calls to `fn`. The second element is always zero for call compatibility with `optim`.

convergence An integer code. `0` indicates successful convergence. Possible error codes are

`1` indicates that the iteration limit `maxit` had been reached.

message Always set to `NULL`, provided for call compatibility with `optim`.

diagnostic List containing diagnostic information. Possible elements are:

sigma Vector containing the step size σ for each iteration.

eigen $d \times niter$ matrix containing the principle components of the covariance matrix C.

pop An $d \times \mu \times niter$ array containing all populations. The last dimension is the iteration and the second dimension the individual.

value A $niter \times \mu$ matrix containing the function values of each population. The first dimension is the iteration, the second one the individual.

These are only present if the respective diagnostic control variable is set to `TRUE`.

Author(s)

Olaf Mersmann <olafm@statistik.tu-dortmund.de> and David Arnu <david.arnu@tu-dortmun.de>

References

Hansen, N. (2006). The CMA Evolution Strategy: A Comparing Review. In J.A. Lozano, P. Larranga, I. Inza and E. Bengoetxea (eds.). Towards a new evolutionary computation. Advances in estimation of distribution algorithms. pp. 75-102, Springer

See Also

`extract_population`

`extract_population` *Extract the iter-th population...*

Description

Extract the `iter`-th population

Usage

```
extract_population(res, iter)
```

Arguments

`res`	A `cma_es` result object.
`iter`	Which population to return.

Details

Return the population of the `iter`-th iteration of the CMA-ES algorithm. For this to work, the populations must be saved in the result object. This is achieved by setting `diag.pop=TRUE` in the `control` list. Function values are included in the result if present in the result object.

Value

A list containing the population as the `par` element and possibly the function values in `value` if they are present in the result object.

`f_rand` *Random function...*

Description

Random function

Usage

```
f_rand(x)
```

Arguments

`x`	parameter vector.

Details

$$f(x) = runif(1)$$

Author(s)

Olaf Mersmann <olafm@statistik.tu-dortmund.de>

`f_rastrigin` *Rastrigin function...*

Description

Rastrigin function

Usage

```
f_rastrigin(x)
```

Arguments

`x`	parameter vector.

Author(s)

David Arnu <david.arnu@tu-dortmund.de>

`f_rosenbrock` *Rosenbrock function...*

Description

Rosenbrock function

Usage

```
f_rosenbrock(x)
```

Arguments

`x`	parameter vector.

Author(s)

David Arnu <david.arnu@tu-dortmund.de>

`f_sphere` *Sphere function...*

Description

Sphere function

Usage

```
f_sphere(x)
```

Arguments

`x`	parameter vector.

Details

$$f(x) = x'x$$

`rotate_function` *Create a rotated test function...*

Description

Create a rotated test function

Usage

```
rotate_function(f, M)
```

Arguments

`f`	test function.
`M`	orthogonal square matrix defining the rotation.

Details

Returns a new rotated test function defined as

$$g(x) = f(Mx).$$

Value

The rotated test function.

Author(s)

Olaf Mersmann <olafm@statistik.tu-dortmund.de>

`shift_function` *shift_function*

Description

Returns a new function

$$g(x) = f(x - offset).$$

Usage

```
shift_function(f, offset)
```

Arguments

`f`	test function
`offset`	offset.

Value

The shifted test function.

Author(s)

Olaf Mersmann `<olafm@statistik.tu-dortmund.de>`

Index