

The DEoptim Package

February 16, 2008

Version 1.2-1

Date 2007-09-12

Title Differential Evolution Optimization

Author David Ardia <david.ardia@unifr.ch>

Maintainer David Ardia <david.ardia@unifr.ch>

Depends R (>= 2.2.0)

Description This package provides the DEoptim function which performs Differential Evolution Optimization (evolutionary algorithm).

License GPL version 2 or newer

URL <http://perso.unifr.ch/david.ardia>

R topics documented:

DEoptim-methods	1
DEoptim	2
Index	6

DEoptim-methods *DEoptim-methods*

Description

Methods for DEoptim objects.

Usage

```
## S3 method for class 'DEoptim':  
summary(object, ...)  
## S3 method for class 'DEoptim':  
plot(x, plot.type = c("bestmemit", "bestvalit"), ...)
```

Arguments

<code>object, x</code>	An object of class <code>DEoptim</code> ; usually, a result of a call to <code>DEoptim</code> .
<code>plot.type</code>	Should we plot the best member at each iteration or the best value at each iteration?
<code>...</code>	Further arguments passed to or from other methods.

Author(s)

David Ardia <david.ardias@unifr.ch>

Examples

```
## Rosenbrock Banana function
Rosenbrock <- function(x) {
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}

lower <- c(-10,-10)
upper <- -lower
r <- DEoptim(Rosenbrock, lower, upper)
summary(r) ## print summary of the DEoptim object

par(mfrow = c(1,2))
plot(r, type = 'b') ## plot the best members
plot(r, plot.type = "bestvalit", type = 'b', col = 'blue') ## and the best values
```

DEoptim

Differential Evolution Optimization

Description

Performs evolutionary optimization via the Differential Evolution algorithm.

Usage

```
DEoptim(FUN, lower, upper, control = list(), ...)
```

Arguments

<code>FUN</code>	A function to be minimized, with first argument the vector of parameters over which minimization is to take place. It should return a scalar result. NA and NaN values are not allowed.
<code>lower, upper</code>	Bounds on the variables.
<code>control</code>	A list of control parameters. See <i>*Details*</i> .
<code>...</code>	Further arguments to be passed to <code>FUN</code> .

Details

DEoptim performs minimization of FUN.

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

VTR The value to be reached. The optimization process will stop if either the maximum number of iterations `itermax` is reached or the best parameter vector `bestmem` has found a value $FUN(\text{bestmem}) \leq VTR$. Default to `-Inf`.

itermax The maximum iteration (population generation) allowed. Default is 200.

NP Number of population members. Default to 50.

F Stepsize from interval [0,2]. Default to 0.8.

CR Crossover probability from interval [0,1]. Default to 0.5.

initial An initial population used as a starting population in the optimization procedure. Maybe useful to speed up the convergence. Defaults to `NULL`.

strategy Defines the binomial DE-strategy used in the optimization procedure:

- 1 best/1
- 2 rand/1
- 3 rand-to-best/1
- 4 best/2
- 5 rand/2

By default `strategy` is 2. See references below for details.

refresh The frequency of reports. Default to every 10 iterations.

digits The number of digits to print when printing numeric values at each iteration. Default to 4.

Value

A list of lists of the class `DEoptim`.

list `optim` contains the followings:

`bestmem`: the best set of parameters found.
`bestval`: the value of FUN corresponding to `bestmem`.
`nfeval`: number of function evaluations.
`iter`: number of procedure iterations.

list `member` contains the followings:

`lower`: the lower boundary.
`upper`: the upper boundary.
`bestvalit`: the best value of FUN at each iteration.
`bestmemit`: the best member at each iteration.
`pop`: the population generated at the last iteration.

Note

DEoptim is a R-vectorized variant of the Differential Evolution algorithm initially developed by Rainer Storn (storn@icsi.berkeley.edu), International Computer Science Institute (ICSI), 1947 Center Street, Suite 600, Berkeley, CA 94704.

If you experience misconvergence in the optimization process you usually have to increase the value for NP, but often you only have to adjust F to be a little lower or higher than 0.8. If you increase NP and simultaneously lower F a little, convergence is more likely to occur but generally takes longer, i.e. DEoptim is getting more robust (there is always a convergence speed/robustness tradeoff).

DEoptim is much more sensitive to the choice of F than it is to the choice of CR. CR is more like a fine tuning element. High values of CR like CR=1 give faster convergence if convergence occurs. Sometimes, however, you have to go down as much as CR=0 to make DEoptim robust enough for a particular problem.

The R-adaptation DEoptim has properties which differ from the original DE version:

1. The random selection of vectors is performed by shuffling the population array. Hence a certain vector cannot be chosen twice in the same term of the perturbation expression.
2. Due to the vectorized expressions DEoptim executes fairly fast.
3. The parameters are constrained within boundaries.
4. An initial population can be given as a starting point for the DE-optimization. This may speed up the convergence if the optimization procedure has to be run many times for slightly different data sets.

To perform a maximization (instead of minimization) of a given function, simply define a new function which is the opposite of the function to maximize and apply DEoptim to it.

To integrate additional constraints on the parameters x of FUN(x), for instance $x[1] + x[2]^2 < 2$, integrate the constraint within the function to optimize, for instance:

```
FUN <- function(x) {
  if (x[1] + x[2]^2 < 2) {
    r <- Inf
  } else {
    ...
  }
  return(r)
}
```

Note that DEoptim stops if any NA or NaN value is obtained. You have to redefine your function to handle these values (for instance, set NA to Inf in your objective function).

Author(s)

David Ardia (david.ardia@unifr.ch) for the R-port; Rainer Storn (storn@icsi.berkeley.edu) for the Differential Evolution algorithm.

References

Differential Evolution homepage :

<http://www.icsi.berkeley.edu/~storn/code.html>

Some useful books:

Price, K. V., Storn, R. M. and Lampinen J. A. (2005) *Differential Evolution - A Practical Approach to Global Optimization*. Springer. ISBN 3540209506.

Nocedal, J. and Wright, S. J. (1999) *Numerical Optimization*. Springer. ISBN 0387987932.

See Also

[DEoptim-methods](#) for methods on DEoptim object; [optim](#) or [constrOptim](#) for constrained optimization.

Examples

```
## Rosenbrock Banana function
Rosenbrock <- function(x) {
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}

lower <- c(-10,-10)
upper <- -lower
DEoptim(Rosenbrock, lower, upper)
DEoptim(Rosenbrock, lower, upper,
  control = list(NP = 100, refresh = 1))
DEoptim(Rosenbrock, lower, upper,
  control = list(NP = 50, itermax = 200, F = 1.5,
  CR = 0.2, refresh = 1))
DEoptim(Rosenbrock, lower, upper,
  control = list(NP = 80, itermax = 400, F = 1.2,
  CR = 0.7, refresh = 1))

## 'Wild' function, global minimum at about -15.81515
Wild <- function(x)
  10 * sin(0.3*x) * sin(1.3*x^2) +
  0.00001 * x^4 + 0.2 * x + 80
plot(Wild, -50, 50, n = 1000,
  main = "DEoptim minimizing 'Wild function'")
DEoptim(Wild, lower = -50, upper = 50,
  control = list(NP = 50, refresh = 1))
DEoptim(Wild, lower = -50, upper = 50,
  control = list(NP = 50, refresh = 1, digits = 8))
```

Index

*Topic **methods**

DEoptim-methods, 1

*Topic **nonlinear**

DEoptim, 2

*Topic **optimize**

DEoptim, 2

constrOptim, 5

DEoptim, 2, 2

DEoptim-methods, 5

DEoptim-methods, 1

optim, 5

plot.DEoptim(*DEoptim-methods*), 1

summary.DEoptim

(*DEoptim-methods*), 1