Package 'lbfgsb3'

February 20, 2015

Type Package

Title Limited Memory BFGS Minimizer with Bounds on Parameters

Version 2015-2.13

Date 2015-02-13

Maintainer John C Nash <nashjc@uottawa.ca>

Description Interfacing to Nocedal et al. L-BFGS-B.3.0 (2011) limited memory BFGS minimizer with bounds on parameters.

License GPL-2

Depends R (>= 2.15.0), numDeriv

Author John C Nash [aut, cre],
Ciyou Zhu [aut],
Richard Byrd [aut],
Jorge Nocedal [aut],
Jose Luis Morales [aut]

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-02-18 23:34:54

R topics documented:

`lbfgsb3` *Interfacing wrapper for the Nocedal - Morales LBFGSB3 (Fortran) limited memory BFGS solver.*

Description

This package is an

Usage

```
lbfgsb3(prm, fn, gr=NULL, lower=-Inf, upper=Inf,
      control=list(), ...)
```

Arguments

prm	A parameter vector which gives the initial guesses to the parameters that will minimize fn. This can be named, for example, we could use prm=c(b1=1, b2=2.345, b3=0.123)
fn	A function that evaluates the objective function to be minimized.
gr	If present, a function that evaluates the gradient vector for the objective function at the given parameterscomputing the elements of the sum of squares function at the set of parameters start.
lower	Lower bounds on the parameters. If a single number, this will be applied to all parameters. Default -Inf.
upper	Upper bounds on the parameters. If a single number, this will be applied to all parameters. Default Inf.
control	An optional list of control settings. See below in details.
...	Any data needed for computation of the objective function and gradient.

Details

See the notes below for a general appreciation of this package.

The list of controls for the algorithm contains:

trace an integer which if 0 (default) causes no intermediate output in the R section of the code, otherwise some diagnostic information

iprint an integer value which if < 0 (-1L default) causes no intermediate output during the running of the Fortran code for lbfgsb3. Other values (see the code lbfgsb.f) give varying amounts of information.

femax NOT YET ACTIVE - Maximum function evaluations.

gemax NOT YET ACTIVE - Maximum number of gradient evaluations.

The output items include the following (not fully edited at 150121).

info <- list(task = task, itask = itask, lsave = lsave, icsave = icsave, dsave = dsave, isave = isave)

icsave is a working integer

lsave is a logical working array of dimension 4. On exit with 'task' = NEW_X, the following information is available: If lsave(1) = .true. then the initial X has been replaced by its projection in the feasible set; If lsave(2) = .true. then the problem is constrained; If lsave(3) = .true. then each variable has upper and lower bounds;

isave is an integer working array of dimension 44. On exit with 'task' = NEW_X, the following information is available: isave(22) = the total number of intervals explored in the search of Cauchy points; isave(26) = the total number of skipped BFGS updates before the current iteration; isave(30) = the number of current iteration; isave(31) = the total number of BFGS updates prior the current iteration; isave(33) = the number of intervals explored in the search of Cauchy point in the current

iteration; isave(34) = the total number of function and gradient evaluations; isave(36) = the number of function value or gradient evaluations in the current iteration; if isave(37) = 0 then the subspace argmin is within the box; if isave(37) = 1 then the subspace argmin is beyond the box; isave(38) = the number of free variables in the current iteration; isave(39) = the number of active constraints in the current iteration; n + 1 - isave(40) = the number of variables leaving the set of active constraints in the current iteration; isave(41) = the number of variables entering the set of active constraints in the current iteration.

dsave is a double precision working array of dimension 29. On exit with 'task' = NEW_X, the following information is available: dsave(1) = current 'theta' in the BFGS matrix; dsave(2) = f(x) in the previous iteration; dsave(3) = factr*epsmch; dsave(4) = 2-norm of the line search direction vector; dsave(5) = the machine precision epsmch generated by the code; dsave(7) = the accumulated time spent on searching for Cauchy points; dsave(8) = the accumulated time spent on subspace minimization; dsave(9) = the accumulated time spent on line search; dsave(11) = the slope of the line search function at the current point of line search; dsave(12) = the maximum relative step length imposed in line search; dsave(13) = the infinity norm of the projected gradient; dsave(14) = the relative step length in the line search; dsave(15) = the slope of the line search function at the starting point of the line search; dsave(16) = the square of the 2-norm of the line search direction vector.

Value

A list of the following items

`prm`	A vector giving the parameter values at the supposed solution.
`f`	The value of the objective function at this set of parameters.
`g`	An estimate of the gradient of the objective at the solution.
`info`	A structure containing information on the disposition of the computation on return. See Details.

Note

This package is a wrapper to the Fortran code released by Nocedal and Morales. This poses several difficulties for an R package. While the `.Fortran()` tool exists for the interfacing, we must be very careful to align the arguments with those of the Fortran subroutine, especially in type and storage.

A more annoying task for interfacing the Fortran code is that Fortran WRITE or PRINT statements must all be replaced with calls to special R-friendly output routines. Unfortunately, the Fortran is full of output statements. Worse, we may wish to be able to suppress such output, and there are thus many modifications to be made. This means that an update of the original code cannot be simply plugged into the R package `src` directory.

Finally, and likely because L-BFGS-B has a long history, the Fortran code is far from well-structured. For example, the number of function and gradient evaluations used is returned as the 34'th element of an integer vector. There does not appear to be an easy way to stop the program after some maximum number of such evaluations have been performed.

On the other hand, the version of L-BFGS-B in `optim()` is a `C` translation of a now-lost Fortran code. It does not implement the improvements Nocedal and Morales published in 2011. Hence, despite its deficiencies, this wrapper has been prepared.

Author(s)

John C Nash <nashjc@uottawa.ca> (of the wrapper and edits to Fortran code to allow R output) Ciyou Zhu, Richard Byrd, Jorge Nocedal, Jose Luis Morales (original Fortran packages)

References

Morales, J. L.; Nocedal, J. (2011). "Remark on 'algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization' ". ACM Transactions on Mathematical Software 38: 1.

Byrd, R. H.; Lu, P.; Nocedal, J.; Zhu, C. (1995). "A Limited Memory Algorithm for Bound Constrained Optimization". SIAM J. Sci. Comput. 16 (5): 1190-1208.

Zhu, C.; Byrd, Richard H.; Lu, Peihuang; Nocedal, Jorge (1997). "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization". ACM Transactions on Mathematical Software 23 (4): 550-560.

See Also

Packages optim and optimx.

Examples

```
cat("Examples are to be added\n")
cat("But see the tests\n")
```

Index