Package 'optimParallel'

October 16, 2018

Type Package

Title Parallel Versions of the Gradient-Based optim() Methods

Version 0.7-4

Date 2018-10-15

Author Florian Gerber

Maintainer Florian Gerber <florian.gerber@math.uzh.ch>

Description Provides parallel versions of the gradient-based optim() methods. The main function of the package is optimParallel(), which has the same usage and output as optim(). Using optimParallel() can significantly reduce the optimization time.

License GPL (>= 2)

URL https://git.math.uzh.ch/florian.gerber/optimParallel

BugReports https://git.math.uzh.ch/florian.gerber/optimParallel/issues

Depends R (>= 3.1), stats, parallel

Suggests R.rsp, roxygen2, spam, microbenchmark, testthat, ggplot2, numDeriv

VignetteBuilder R.rsp

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-15 22:40:02 UTC

R topics documented:

optimParallel *parallel version of* `optim`

Description

The function provides parallel versions of the gradient-based `optim` methods `"L-BFGS-B"`, `"BFGS"`, and `"CG"`. If the evaluation of the function `fn` takes more than 0.05 seconds, `optimParallel` can significantly reduce the optimization time. For a p-parameter optimization based on `"L-BFGS-B"`, the speed increase is about factor $1+2p$ when no analytic gradient is specified and $1+2p$ processor cores are available.

Usage

```
optimParallel(par, fn, gr = NULL, ..., method = c("L-BFGS-B", "BFGS", "CG"),
  lower = -Inf, upper = Inf, control = list(), hessian = FALSE,
  parallel = list())
```

Arguments

`par`	see the documentation of `optim`.
`fn`	see the documentation of `optim`.
`gr`	see the documentation of `optim`.
`...`	see the documentation of `optim`. Note that depending on the chosen cluster type for parallel execution, the `.GlobalEnv` of the R processes in the cluster contain different R objects compared to the main R process. In that case, it may be necessary to add all R object required by `fn` and `gr` here in order to pass them to the R processes in the cluster.
`method`	parallel versions of the gradient-based methods `"L-BFGS-B"` (default), `"BFGS"`, and `"CG"` of `optim` are available. The recommended method is `"L-BFGS-B"` because it triggers one (approximate) gradient evaluation per iteration, which best fits the implemented parallel processing scheme. See the documentation of `optim` for information on the methods. If another method is specified, all arguments are directly passed to `optim`.
`lower`	see the documentation of `optim`.
`upper`	see the documentation of `optim`.
`control`	see the documentation of `optim`.
`hessian`	see the documentation of `optim`.
`parallel`	is a list of additional control parameters and can supply any of the following components:

`cl` an object of class `"cluster"` specifying the cluster to be used for parallel execution. See `makeCluster` for more information. If the argument is not specified or `NULL`, the default cluster is used. See `setDefaultCluster` for information on how to set up a default cluster.

`forward` logical vector of length 1. If `FALSE` (default when loading the package), a numeric central difference approximation of the gradient defined as $(fn(x+\epsilon)-fn(x-\epsilon))/(2\epsilon)$ is used, which corresponds to the approximation used in `optim`. If `TRUE`, a numeric forward difference approximation of the gradient essentially defined as $(fn(x+\epsilon)-fn(x))/\epsilon$ is used. This reduces the number of function calls from $1+2p$ to $1+p$ and can be useful if the number of available cores is smaller than $1+2p$ and if the memory limit is reached.

`loginfo` logical vector of length 1 with default value `FALSE` when loading the package. If `TRUE`, additional log information containing the evaluated parameters as well as return values of `fn` and `gr` is returned.

Details

`optimParallel` is a wrapper to `optim` and relies on the lexical scoping mechanism of R and the R package **parallel** to evaluate `fn` and its (approximate) gradient in parallel.

Some default values of the argument `parallel` can be set via
`options("optimParallel.forward", "optimParallel.loginfo")`.

Value

Same as the return value of `optim`. See the documentation thereof for more information.
If a gradient-based method is specified and `parallel=list(loginfo=TRUE)`, additional log information containing the evaluated parameters as well as the return values of `fn` and `gr` is returned.

Notes

1. If `fn` or `gr` depend on functions or methods from loaded packages, it may be necessary to explicitly load those packages in all processes of the cluster. For `cl` of class `"cluster"` one can use `clusterEvalQ(cl, search())` to check whether all required packages are on the search paths of all processes. If, for example, the R package **spam** is required and missing on those search paths, it can be added via `clusterEvalQ(cl, library("spam"))`.
2. If `fn` or `gr` depend on functions or objects defined in the current R session, it may be necessary to pass them to `optimParallel` via the `...` argument. Alternatively, they can be made available to the R processes in the cluster via `clusterEvalQ`.
3. Using parallel R code inside `fn` and `gr` may not work, because this results in nested parallel processing.
4. Using `optimParellel` with n parallel processes increases the memory usage by about factor n compared to a call to `optim`. If the memory limit is reached this may severely slowdown the optimization. Strategies to reduce memory usage are (1) kill all unused processes on the computer, (2) revise the code of `fn` and/or `gr` to reduce its memory usage, and (3) reduce the number of parallel processes by specifying the argument `parallel=list(forward=TRUE)` and/or setting up a cluster with less parallel processes.

Issues and bug report

A list of known issues of `optimParallel` can be found at https://git.math.uzh.ch/florian.gerber/optimParallel/issues. Please report issues not listed there to `<florian.gerber@math.uzh.ch>`.

Do not forget to include an R script reproducing the issue and the output of `sessionInfo()`.

Author(s)

Florian Gerber, <florian.gerber@math.uzh.ch>, https://user.math.uzh.ch/gerber.

References

F. Gerber, R. Furrer (2018) optimParallel: an R Package Providing Parallel Versions of the Gradient-Based Optimization Methods of optim(). ArXiv e-prints. URL http://arxiv.org/abs/1804.11058. Also available as vignette of this package `vignette("optimParallel")`.

See Also

`optim`, `makeCluster`, `setDefaultCluster`, `stopCluster`, `detectCores`.

Examples

```
negll <- function(par, x, sleep=0, verbose=TRUE){
    if(verbose)
        cat(par, "\n")
   Sys.sleep(sleep)
   -sum(dnorm(x=x, mean=par[1], sd=par[2], log=TRUE))
}
set.seed(13); x <- rnorm(1000, 5, 2)

cl <- makeCluster(2)     # set the number of processor cores
setDefaultCluster(cl=cl) # set 'cl' as default cluster

optimParallel(par=c(1,1), fn=negll, x=x,
              method = "L-BFGS-B", lower=c(-Inf, .0001))

optimParallel(par=c(1,1), fn=negll, x=x,
              method = "L-BFGS-B", lower=c(-Inf, .0001),
              parallel=list(loginfo=TRUE))

setDefaultCluster(cl=NULL); stopCluster(cl)

## default values of the argument 'parallel':
options("optimParallel.forward", "optimParallel.loginfo")

## Not run:
## - use all avilable processor cores
## - return cat() output to R prompt
##   (may have issues on Windows)
if(tolower(.Platform$OS.type) != "windows"){
    cl <- makeCluster(spec=detectCores(), type="FORK", outfile="")
} else
    cl <- makeCluster(spec=detectCores(), outfile="")
setDefaultCluster(cl=cl)

## return log information
```

```
options(optimParallel.loginfo=TRUE)

## stop if change of f(x) is smaller than 0.01
control <- list(factr=.01/.Machine$double.eps)

optimParallel(par=c(1,1), fn=negll, x=x, sleep=.5,
              verbose=TRUE, method="L-BFGS-B",
              lower=c(-Inf, .0001), control=control)
## each step invokes 5 parallel calls to negll()

optimParallel(par=c(1,1), fn=negll, x=x, sleep=.5,
              method ="L-BFGS-B", lower=c(-Inf, .0001),
              control=control,
              parallel=list(forward=TRUE))
## each step invokes 3 parallel calls to negll()

setDefaultCluster(cl=NULL); stopCluster(cl)
## End(Not run)
```

Index