Chp 2: Optimization

Created By Jennifer Hoeting (Demo Student) on Wednesday, August 27, 2014 7:45:58 AM MDT
last modified by Josh Hewitt on Monday, September 8, 2014 12:13:12 PM MDT

Comparing and Contrasting the optimize() and nlminb() functions in R.

optimize(f, interval, maximum=FALSE, tol=.Machine$double.eps^.25)

optimize() takes a function, f, and finds the point at which it acheives its minimum (by default). It will find the maximum of this function, f, if you set maximum=TRUE, although this is equivalent to minimizing -f. If f has more than one argument, it will only be optimized with respect to its first argument. It should also be noted that the convergence order is superlinear (1.324) if f has a positive second derivative at its minimum value, although the number of iterations is not given in the output of optimize().

The interval argument is a vector of two values, the lower bound and the upper bound, in which the value for the argument that optimizes f is assumed to be.

The tolerance, tol, is the precision of the optimized argument. By default, it is equal to .Machine$double.eps^.25, which itself is equal to 0.0001220703. So, by default, optimize() will return the parameter with a precison of about four decimal places.

optimize() will return a list with two values: minimum (or maximum), the value of the argument over which f is optimized, and objective, the value of f evaluated at the argument which optimizes it.

Using optimize() with the example from chapter two in the book:

```
> f=function(x){log(x)/(1+x)}
> optimize(f, c(1, 5), maximum=T, tol=.00001)
$maximum
[1] 3.591123

$objective
[1] 0.2784645
```

nlminb(start, objective, control, lower, upper)

nlminb() takes a function, objective, and finds values for the parameters of this function at which the objective function acheives its minimum value. Unlike optimize(), nlminb() can be used to optimize functions with mutiple arguments. nlminb() also cannot be used to find a point which maximizes the objective function, so to find the maximum, we must minimize the negative objective function. Based on some testing, it appreas the convergence order is superlinear and close to 2.

The start argument is a vector of starting points for each of the parameters of the objective function.

control is a list of control paremters such as the maximum number of iterations (iter.max) and relative tolerance (rel.tol), which defaults to 1e-10.

lower and upper are the bounds for constrained optimization, which can be especially useful if there are many local minimums. lower is negative infinity by default and upper is infinity by default.

nlminb() returns a list with six arguments. par is (are) the value(s) of the parameter(s) that optimize(s) the objectice function. objective is the the value of the objective function evaluated at the arguments which optimize it. convergence, if 0, indicated a successful convergence. iterations is the number of iterations needed to achieve absolute convergence. evaluations tells the number of times the objective function and its derivative were evaluated. Finally, message gives some additonal information, often displaying the number of iterations needed to

achieve relative convergence.

Using nlimb() with the example from chapter two in the book:

```
> obj=function(x){-log(x)/(1+x)}
> nlminb(3,obj)
$par[1] 3.591123
$objective
[1] -0.2784645
$convergence
[1] 0
$iterations
[1] 7
$evaluations
function gradient
9        9
$message
[1] "relative convergence (4)"
```

Last note: when doing multivariate optimization using nlminb(), make the objective function a function of one variable, say x, and then put the differnet levels of x into the function.

Example:

```
> g=function(x){(((x[1]^2)+x[2]-11)^2)+(x[1]+(x[2]^2)-7)^2}
> nlminb(start= c(0, 0), g)
$par
[1] 3 2
$objective
[1] 6.540553e-21
$convergence
[1] 0
$iterations
[1] 10
$evaluations
function gradient
14       24
$message
[1] "X-convergence (3)"
```

Comparing optim(), nlm(), ucminf() (and optimx()) in R.

Josh Hewitt.

optim

Optimization method(s): Optim is a wrapper function for the Nelder-Mead, BFGS, constrained BFGS, conjugate-gradient, Brent, and simulated annealing methods. Users may choose which method they wish to apply. Except for Brent's method, these methods are all capable of optimizing multivariate functions.

Convergence criteria: By default, optim uses the relative convergence criteria but users may choose to apply the absolute convergence criteria instead.

nlm

Optimization method(s): Nlm is a little mysterious since it claims to use a “Newton-type” algorithm and provides few elaborating details. I believe this method uses a Newton-like approach to determine a direction to move in and a line-search to determine step sizes. Although users can give nlm gradient and Hessian functions to use

during the minimization, neither the gradient nor the Hessian are required. Instead, nlm seems to either approximate them numerically or derive them symbolically. ...Very mysterious.

Convergence criteria: Nlm will converge if relative convergence criteria are satisfied either for the function's argmin or for its gradient; i.e., the algorithm will converge if the estimated minimizer appears to stabilize or if the gradient appears to be 0. In addition to exceeding a maximum iteration limit, this method will also fail to converge if the algorithm does not improve its estimate of the function's minimizer at any step.

ucminf

Optimization method(s): ucminf is similar to nlm. At each iteration, ucminf uses a BFGS quasi-Newton step to determine a direction in which to proceed with the optimization and a line search to determine a step size. The method uses finite differences to approximate the gradient and Hessian if the user does not provide these functions.

Convergence criteria: Like nlm, ucminf will converge if relative convergence criteria are satisfied for the function's argmin or for its gradient. Similarly, the method will also fail to converge if the maximum iteration limit is exceeded or the algorithm cannot improve its estimate of the function's minimizer at any step.

optimx

This is by special request from Miranda.

Optimization method(s): optimx extends optim by (mostly) adding additional optimization methods to the wrapper function optim provides. optimx removes support for simulated annealing, but adds support for many other optimization functions available in R: nlm, nlminb, ucminf, and seven others (see help file for full details).

Why? The optimx help file essentially says it is designed to satisfy two practical programming goals: 1) to further standardize function calls to optimization algorithms so that users can more quickly switch between different algorithms, and 2) to let users run several optimization algorithms with one call to the wrapper function (they note, in comparison, that optim only lets users run one method per function call).