Optimization using R

Xiangyun Huang

China University of Mining & Technology,Beijing

July 8, 2016

Contents

1 无约束优化

2 非线性优化求解器

3 非线性混合整数规划

4 多目标规划

5 通用求解器

stats 包

- optim
- optimize
- nlm
- nlminb
- constrOptim

```r
library(stats)
optim()
optimize()
nlm()
nlminb()
constrOptim()
```

optim 基于 Nelder-Mead(单纯型法)、quasi-Newton(拟牛顿法) 和 CG(共轭梯度法)

```
optim(par, fn, gr = NULL, ...,
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN",
                 "Brent"),
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)

optimHess(par, fn, gr = NULL, ..., control = list())
```

1. BFGS 算法 (属于 quasi-Newton 算法) 分别由 Broyden，Fletcher，Goldfarb 和 Shanno 在 1970 年同时提出，算法使用函数值及梯度优化
2. L-BFGS-B 算法允许 box constraints(箱式约束)，即每个变量可以有上下界限制
3. SANN 算法指 simulated annealing(模拟退火) 算法，属于随机全局优化方法。仅使用函数值，优点在于适用目标函数不可微的情况，缺点是效率较低
4. Brent 算法只适用于一维搜索问题，如用于 optimize，uniroot 和 mle 函数

非线性无约束优化

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

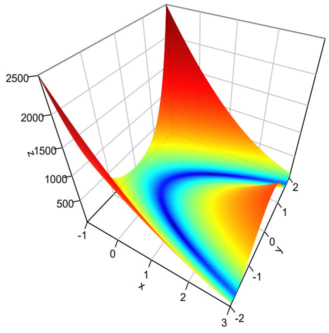


图 1: Rosenbrock 函数

```
fr <- function(x) {
  100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2
}
library(numDeriv)
grr <- function(x) {
  grad(fr,c(x[1],x[2]))  #  gradient
}
optim(c(-1.2, 1), fr, grr, method = "BFGS")

$par
[1] 1 1

$value
[1] 9.595012e-18

$counts
function gradient
     110       43

$convergence
[1] 0

$message
NULL
```

大规模非线性无约束优化

$$\min f(X) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

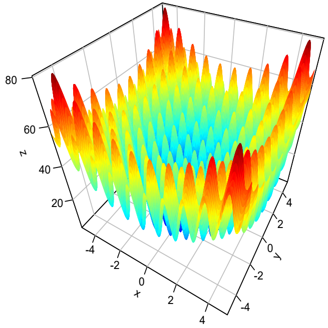


图 2: Radistrigin 函数

```r
library(Rdonlp2)
fn <- function(x) {
  sum(x^2-10*cos(2*pi*x)+10)
  }
result<-donlp2(rep(-500,100),fn)
```

100 维空间中，该函数极小值点特别多，常用于测试各类优化算法，Rdonlp2 包可以在不到 2 秒内求解[a]

[a]注：i7-4710MQ 四核 2.5GHz，内存 8G

- 非线性方程 (组) 求根 uniroot 和 multiroot(rootSolve 包)
- hessian 黑塞矩阵、jacobian 雅可比矩阵、grad 梯度向量

非线性方程求根

```
optimHess(c(-1,1),fr) # stats

         [,1] [,2]
[1,] 802.0008  400
[2,] 400.0000  200

grad(fr,c(-1,1)) # numDeriv

[1] -4  0

jacobian(fr,c(-1,1)) # numDeriv

     [,1] [,2]
[1,]   -4    0

hessian(fr,c(-1,1))  # numDeriv

     [,1] [,2]
[1,]  802  400
[2,]  400  200
```

```
f <- function(x) x^3+3*x+1
# multiroot(f,start = 1 )
uniroot(f,c(-10,10))

$root
[1] -0.3221734

$f.root
[1] 3.970168e-05

$iter
[1] 12

$init.it
[1] NA

$estim.prec
[1] 6.103516e-05

# library(rootSolve)
# uniroot.all()
```

非线性方程组求根：基于 Newton-Raphson 方法

```
library(rootSolve)
equations <- function(x){
  f1 <- x[1]^2-x[2]-1
  f2 <- x[1]^2-4*x[1]+x[2]^2-x[2]+3.25
  c(f1,f2)
}
(ss <- multiroot(f = equations, start = c(0, 0)))

$root
[1] 1.0673461 0.1392277

$f.root
[1] 6.027845e-12 4.951728e-11

$iter
[1] 7

$estim.precis
[1] 2.777256e-11

# (ss2 <- multiroot(f = equations, start = c(2, 2)))
equations(ss$root)

[1] 6.027845e-12 4.951728e-11
```

nleqslv 包

基于 Broyden 秩 1 修正和 Newton 法，采用全局线搜索 (cubic, quadratic or geometric) 和信赖域方法 (double dogleg, Powell single dogleg or Levenberg-Marquardt type) 求解非线性方程组，雅可比矩阵可以奇异或者病态

```r
# R. Baker Kearfott, Some tests of Generalized Bisection,
# ACM Transactions on Methematical Software, Vol. 13, No. 3, 1987, pp 197-220
# A high-degree polynomial system (section 4.3 Problem 12)
# There are 12 real roots (and 126 complex roots to this system!)
library(nleqslv)
hdp <- function(x) {
  f <- numeric(length(x))
  f[1] <- 5 * x[1]^9 - 6 * x[1]^5 * x[2]^2 + x[1] * x[2]^4 + 2 * x[1] * x[3]
  f[2] <- -2 * x[1]^6 * x[2] + 2 * x[1]^2 * x[2]^3 + 2 * x[2] * x[3]
  f[3] <- x[1]^2 + x[2]^2 - 0.265625
  f
}
N <- 40 # at least to find all 12 roots
set.seed(123)
xstart <- matrix(runif(3*N,min=-1,max=1), N, 3) # N initial guesses, each of length 3
ans <- searchZeros(xstart,hdp, method="Broyden",global="dbldog")
ans$x
```

一维无约束优化：optimize 基于 Brent 算法，默认求极小

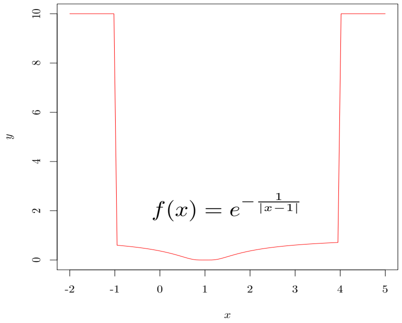


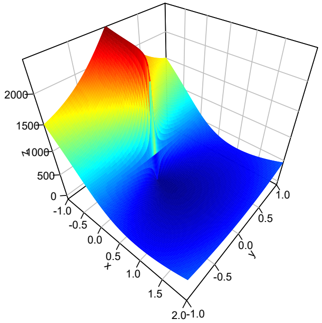
图 3: $f(x)$ 函数

- optimise 与 optimize 是一回事！
- 糟糕的区间可能得到错误结果，结合图示比较好

```
f  <- function(x) {
  ifelse(x > -1,
         ifelse(x < 4,
                exp(-1/abs(x - 1)),
                10), 10)
}
fp <- function(x) {  f(x) }
# doesn't see the minimum
optimize(fp, c(-4, 20))

$minimum
[1] 19.99995

$objective
[1] 10

optimize(fp, c(-7, 20))   # ok

$minimum
[1] 0.9992797

$objective
[1] 0
```

多维无约束优化：nlm 基于牛顿型算法，默认求极小

```
theta <- function(x1,x2) atan2(x2, x1)/(2*pi)
f <- function(x) {
     f1 <- 10*(x[3] - 10*theta(x[1],x[2]))
     f2 <- 10*(sqrt(x[1]^2+x[2]^2)-1)
     f3 <- x[3]
     return(f1^2+f2^2+f3^2)
}
nlm.f <- nlm(f, c(-1,0,0), hessian = FALSE)
nlm.f

$minimum
[1] 1.238234e-14

$estimate
[1]  1.000000e+00  3.071078e-09 -6.063454e-09

$gradient
[1] -3.755763e-07  3.485884e-06 -2.202372e-06

$code
[1] 2

$iterations
[1] 27
```



```
nlm(f, p, ..., hessian = FALSE,
    typsize = rep(1, length(p)),
    fscale = 1, print.level = 0,
    ndigit = 12, gradtol = 1e-6,
   steptol = 1e-6, iterlim = 100,
    check.analyticals = TRUE)
```

nlminb：求解多维无约束或简单箱式约束优化问题 using PORT routines

```
flb <- function(x){
  p <- length(x)
  sum(c(1, rep(4, p-1))*(x - c(1, x[-p])^2)^2)
}
n=3 ## 3-dimensional box constrained
nlminb(rep(3, n), flb,
       lower = rep(2, n),
       upper = rep(4, n))

$par
[1] 2.000000 2.109094 4.000000

$objective
[1] 16.10591

$convergence
[1] 0

$iterations
[1] 6

$evaluations
function gradient
       7       22

$message
[1] "relative convergence (4)"
```

```
n=3
# 3-dimensional box constrained
f<-function(x){
  (x[1]-1)^2+
  4*(x[2]-x[1]^2)^2+
  4*(x[3]-x[2]^2)^2
}
nlminb(rep(3, n), f,
       lower = rep(2, n),
       upper = rep(4, n))
```

constrOptim：基于自适应性障碍算法求解线性不等式约束优化问题

```r
constrOptim(theta, f, grad, ui, ci, mu = 1e-04, control = list(),
            method = if(is.null(grad)) "Nelder-Mead" else "BFGS",
            outer.iterations = 100, outer.eps = 1e-05, ...,
            hessian = FALSE)
## from optim
fr <- function(x) {   ## Rosenbrock Banana function
    x1 <- x[1]
    x2 <- x[2]
    100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
grr <- function(x) { ## Gradient of 'fr'
    x1 <- x[1]
    x2 <- x[2]
    c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
       200 *      (x2 - x1 * x1))
}
#  x <= 0.9,  y - x > 0.1
# ui %*% theta - ci >= 0
constrOptim(c(.5,1), fr, grr,
            ui = rbind(c(-1,0), c(-1,1)),
            ci = c(-0.9,0.1))
```

Contents

1 无约束优化

2 非线性优化求解器

3 非线性混合整数规划

4 多目标规划

5 通用求解器

- optimx：对基本优化函数 (除 SANN 算法外) 封装，BB 包
- alabama：增广拉格朗日自适应障碍最小化算法
- Rdonlp2：非线性规划 (约束条件和目标函数都是非线性)
- Rsolnp：基于增广拉格朗日乘子法

alabama、Rdonlp2 和 Rsolnp 目标函数都是连续或者可微的

optimx 包

- General-purpose optimization wrapper function that calls other R tools for optimization.
- 统一了 stats、BB 和 ucminf 包内的函数接口，但是 optimx 函数不调用 optim 函数内的 SANN 算法.

```
library(stats)
library(BB)
library(ucminf) # An Algorithm for unconstrained nonlinear optimization
library(optimx)
```

BB 包

适合大规模非线性方程求根 (BBsolve) 和带箱式约束的非线性优化 (BBoptim)，基于 Barzilai-Borwein 方法实现，不要求目标函数的 Hessian 矩阵

```r
library(BB)
equations <- function(x){
  f1 <- x[1]^2-x[2]-1
  f2 <- x[1]^2-4*x[1]+x[2]^2-x[2]+3.25
  c(f1,f2)
}
p0 <- rep(0, 2)
# dfsane(par = p0, fn = equations, control = list(trace = FALSE))
BBsolve(par = p0, fn = equations) # BBsolve is better
```

BBsolve 是 dfsane 的封装

二项混合泊松分布的参数最大似然估计

```r
poissmix.loglik <- function(p,y) {
  # Log-likelihood for a binary Poisson mixture distribution
  i <- 0:(length(y)-1)
  loglik <- y * log(p[1] * exp(-p[2]) * p[2]^i / exp(lgamma(i+1)) +
            (1 - p[1]) * exp(-p[3]) * p[3]^i / exp(lgamma(i+1)))
  return (sum(loglik) )
 }
# Data from Hasselblad (JASA 1969)
poissmix.dat <- data.frame(death=0:9,freq=c(162,267,271,185,111,61,27,8,3,1))
lo <- c(0,0,0) # lower limits for parameters
hi <- c(1, Inf, Inf) # upper limits for parameters
p0 <- runif(3,c(0.2,1,1),c(0.8,5,8)) # a randomly generated vector of length 3
y <- c(162,267,271,185,111,61,27,8,3,1)
ans1 <- spg(par=p0, fn=poissmix.loglik, y=y,
            lower=lo, upper=hi,
            control=list(maximize=TRUE, trace=FALSE))
ans2 <- BBoptim(par=p0, fn=poissmix.loglik, y=y,
                lower=lo, upper=hi, control=list(maximize=TRUE))
ans2
```

计算最大似然处的黑塞矩阵以及参数的标准差

```r
library(numDeriv)
hess <- hessian(x=ans2$par, func=poissmix.loglik, y=y)
# Note that we have to supplied data vector 'y'
hess
se <- sqrt(diag(solve(-hess)))
se
```

从不同初始值出发尝试寻找全局最大值，实际找的是一系列局部最大值

```
# 3 randomly generated starting values
p0 <- matrix(runif(30, c(0.2,1,1), c(0.8,8,8)), 10, 3, byrow=TRUE)
ans <- multiStart(par=p0, fn=poissmix.loglik, action="optimize",
                y=y, lower=lo, upper=hi, control=list(maximize=TRUE))

# selecting only converged solutions
pmat <- round(cbind(ans$fvalue[ans$conv], ans$par[ans$conv, ]), 4)
dimnames(pmat) <- list(NULL, c("fvalue","parameter 1","parameter 2","parameter 3"))
pmat[!duplicated(pmat), ]
```

Goal: 带非线性约束的非线性优化

alabama 包

Augmented Lagrangian Adaptive Barrier Minimization Algorithm
(增广拉格朗日自适应障碍最小化算法): 可带 (非) 线性 (不) 等式约束

```r
library(alabama)
fn <- function(x) (x[1] + 3*x[2] + x[3])^2 + 4 * (x[1] - x[2])^2
gr <- function(x) {
  g <- rep(NA, 3)
  g[1] <- 2*(x[1] + 3*x[2] + x[3]) + 8*(x[1] - x[2])
  g[2] <- 6*(x[1] + 3*x[2] + x[3]) - 8*(x[1] - x[2])
  g[3] <- 2*(x[1] + 3*x[2] + x[3])
  g
}
heq <- function(x) {
  h <- rep(NA, 1)
  h[1] <- x[1] + x[2] + x[3] - 1
  h
}
heq.jac <- function(x) {
  j <- matrix(NA, 1, length(x))
  j[1, ] <- c(1, 1, 1)
  j
}
```

```
hin <- function(x) {
  h <- rep(NA, 1)
  h[1] <- 6*x[2] + 4*x[3] - x[1]^3 - 3
  h[2] <- x[1]
  h[3] <- x[2]
  h[4] <- x[3]
  h
}
hin.jac <- function(x) {
  j <- matrix(NA, 4, length(x))
  j[1, ] <- c(-3*x[1]^2, 6, 4)
  j[2, ] <- c(1, 0, 0)
  j[3, ] <- c(0, 1, 0)
  j[4, ] <- c(0, 0, 1)
  j
}
set.seed(12)
p0 <- runif(3)
ans <- constrOptim.nl(par=p0, fn=fn, gr=gr, heq=heq,
                      heq.jac=heq.jac, hin=hin, hin.jac=hin.jac)
# Not specifying the gradient and the Jacobians
set.seed(12)
p0 <- runif(3)
ans2 <- constrOptim.nl(par=p0, fn=fn, heq=heq, hin=hin)
```

Rdonlp2 包

先安装 gcc/g++ 编译环境：Rtools [1]

```
install.packages("Rdonlp2", repos="http://R-Forge.R-project.org",type = "source")
library(Rdonlp2)
```

Rtools 的版本与 R 的版本要相容，在 R Console 里运行下面的代码，返回 TRUE，说明安装成功。

```
devtools::find_rtools()

[1] TRUE
```

[1] http://cran.rstudio.com/bin/windows/Rtools/Rtools32.exe

求下列有约束的非线性规划问题

$$\min \ \ z = x^2 \sin y + y^2 \cos x$$
$$s.t. \begin{cases} -100 < x < 100 \\ -100 < y < 100 \\ 1 \leqslant 3x - y \leqslant 3 \\ x + y \geqslant 2 \\ \sin x \cos y \leqslant 0.6 \\ xy = 2 \end{cases}$$

```
p = c(10,10)
par.l= c(-100,-100); par.u = c(100,100)
fn = function(x){
  x[1]^2*sin(x[2])+x[2]^2*cos(x[1])
}
A = matrix(c(1,1,3,-1),2,byrow=TRUE)
lin.l = c(2,1);lin.u = c(+Inf,3)
nlcon1 = function(x){
  x[1]*x[2]
}
nlcon2 = function(x){
  sin(x[1])*cos(x[2])
}
nlin.l = c(2,-Inf)
nlin.u = c(2,0.6)
ret = donlp2(p, fn, par.u=par.u, par.l=par.l,A,lin.l=lin.l,lin.u=lin.u,
             nlin=list(nlcon1,nlcon2), nlin.u=nlin.u, nlin.l=nlin.l)
# $par
# [1] 1.403076 1.425440
# $fx
# [1] 2.287053
# $message
# [1] "KT-conditions satisfied, no further correction computed"
```

Rsolnp 包

General Non-linear Optimization Using Augmented Lagrange Multiplier Method (基于增广拉格朗日乘子法)

```r
# POWELL Problem
library(Rsolnp)
fn1=function(x){
  exp(x[1]*x[2]*x[3]*x[4]*x[5])
}
eqn1=function(x){
  z1=x[1]*x[1]+x[2]*x[2]+x[3]*x[3]+x[4]*x[4]+x[5]*x[5]
  z2=x[2]*x[3]-5*x[4]*x[5]
  z3=x[1]*x[1]*x[1]+x[2]*x[2]*x[2]
  return(c(z1,z2,z3))
}
x0 = c(-2, 2, 2, -1, -1)
powell=solnp(x0, fun = fn1, eqfun = eqn1, eqB = c(10, 0, -1))
```

再来个栗子

```r
library(Rsolnp)
library(parallel)
cl = makePSOCKcluster(2)
gofn = function(dat, n)
{
  x = dat[1:n]
  y = dat[(n+1):(2*n)]
  z = dat[(2*n+1):(3*n)]
  ii = matrix(1:n, ncol = n, nrow = n, byrow = TRUE)
  jj = matrix(1:n, ncol = n, nrow = n)
  ij = which(ii<jj, arr.ind = TRUE)
  i = ij[,1]
  j = ij[,2]
  # Coulomb potential
  potential = sum(1.0/sqrt((x[i]-x[j])^2 + (y[i]-y[j])^2 + (z[i]-z[j])^2))
  potential
}
goeqfn = function(dat, n)
{
  x = dat[1:n]
  y = dat[(n+1):(2*n)]
  z = dat[(2*n+1):(3*n)]
  apply(cbind(x^2, y^2, z^2), 1, "sum")
}
```

```
n = 25
LB = rep(-1, 3*n)
UB = rep( 1, 3*n)
eqB = rep( 1, n)
sp = startpars(pars = NULL, fixed = NULL, fun = gofn ,
               eqfun = goeqfn,eqB = eqB, ineqfun = NULL,
               ineqLB = NULL, ineqUB = NULL, LB = LB,
               UB = UB,distr = rep(1, length(LB)),
               distr.opt = list(), n.sim = 2000,cluster = cl,
               rseed = 100, bestN = 15, eval.type = 2, n = 25)
#stop cluster
stopCluster(cl)
# the last column is the value of the evaluated function
# (here it is the barrier function since eval.type = 2)
print(round(apply(sp, 2, "mean"), 3))
# remember to remove the last column
ans = solnp(pars=sp[1,-76],fun = gofn , eqfun = goeqfn,
            eqB = eqB, ineqfun = NULL,ineqLB = NULL,
            ineqUB = NULL, LB = LB, UB = UB, n = 25)
# should get a value of around 243.8162
```

Contents

1 无约束优化
2 非线性优化求解器
3 非线性混合整数规划
4 多目标规划
5 通用求解器

Nonlinear Mixed Integer Programming

$$\min f(x)$$
$$s.t. \begin{cases} g_L \leq g(x) \leq g_U \\ x_L \leq x \leq x_U \\ x_i \in \mathbb{Z}, i \in I, \\ x_i \in \mathbb{R}, i \notin I. \end{cases}$$

where $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$, $g(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ are twice continuously differentiable convex functions and I is a subset of $\mathbb{Z}_+$

- lpSolve：线性、整数和混合整数规划，目标函数为线性
- Rglpk：读取 MathProg 和 CPLEX 的接口，线性和混合整数规划
- 非线性混合整数规划 MINLP，Bonmin 开源项目 [2]
- 启发式算法 GA

[2]https://projects.coin-or.org/Bonmin

Contents

1 无约束优化

2 非线性优化求解器

3 非线性混合整数规划

4 多目标规划

5 通用求解器

Contents

1 无约束优化

2 非线性优化求解器

3 非线性混合整数规划

4 多目标规划

5 通用求解器

商业优化软件及其 R 包接口

- Gurobi
- LocalSolver(localsolver)
- Lindo(rLindo)
- CPLEX(Rcplex 和 cplexAPI) 可获得免费学术版
- MOSEK(Rmosek) 可获得免费学术版

Thank You