Package ‘NlcOptim’

October 12, 2022

Title Solve Nonlinear Optimization with Nonlinear Constraints

Version 0.6

Author Xianyan Chen <xychen@uga.edu>, Xiangrong Yin <yinxiangrong@uky.edu>

Maintainer Xianyan Chen <xychen@uga.edu>

Description Optimization for nonlinear objective and constraint functions. Linear or nonlinear equality and inequality constraints are allowed. It accepts the input parameters as a constrained matrix.

Depends MASS, R (>= 3.2.2)

LazyData true

Imports quadprog

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-01-18 15:00:31 UTC

R topics documented:

solnl 1

Index **7**

`solnl` *Solve Optimization problem with Nonlinear Objective and Constraints*

Description

Sequential Quatratic Programming (SQP) method is implemented to find solution for general nonlinear optimization problem (with nonlinear objective and constraint functions). The SQP method can be find in detail in Chapter 18 of Jorge Nocedal and Stephen J. Wright's book. Linear or nonlinear equality and inequality constraints are allowed. It accepts the input parameters as a constrained matrix. The function `solnl` is to solve generalized nonlinear optimization problem:

$$minf(x)$$

$$s.t.ceq(x) = 0$$

$$c(x) \leq 0$$

$$Ax \leq B$$

$$Aeqx \leq Beq$$

$$lb \leq x \leq ub$$

Usage

```
solnl(X = NULL, objfun = NULL, confun = NULL, A = NULL, B = NULL,
  Aeq = NULL, Beq = NULL, lb = NULL, ub = NULL, tolX = 1e-05,
  tolFun = 1e-06, tolCon = 1e-06, maxnFun = 1e+07, maxIter = 4000)
```

Arguments

`X`	Starting vector of parameter values.
`objfun`	Nonlinear objective function that is to be optimized.
`confun`	Nonlinear constraint function. Return a `ceq` vector and a `c` vector as nonlinear equality constraints and an inequality constraints.
`A`	A in the linear inequality constraints.
`B`	B in the linear inequality constraints.
`Aeq`	Aeq in the linear equality constraints.
`Beq`	Beq in the linear equality constraints.
`lb`	Lower bounds of parameters.
`ub`	Upper bounds of parameters.
`tolX`	The tolerance in X.
`tolFun`	The tolerance in the objective function.
`tolCon`	The tolenrance in the constraint function.
`maxnFun`	Maximum updates in the objective function.
`maxIter`	Maximum iteration.

Value

Return a list with the following components:

`par`	The optimum solution.
`fn`	The value of the objective function at the optimal point.
`counts`	Number of function evaluations, and number of gradient evaluations.
`lambda`	Lagrangian multiplier.
`grad`	The gradient of the objective function at the optimal point.
`hessian`	Hessian of the objective function at the optimal point.

Author(s)

Xianyan Chen, Xiangrong Yin

References

Nocedal, Jorge, and Stephen Wright. Numerical optimization. Springer Science & Business Media, 2006.

Examples

```
library(MASS)
###ex1
objfun=function(x){
 return(exp(x[1]*x[2]*x[3]*x[4]*x[5]))
}
#constraint function
confun=function(x){
 f=NULL
 f=rbind(f,x[1]^2+x[2]^2+x[3]^2+x[4]^2+x[5]^2-10)
 f=rbind(f,x[2]*x[3]-5*x[4]*x[5])
 f=rbind(f,x[1]^3+x[2]^3+1)
 return(list(ceq=f,c=NULL))
}

x0=c(-2,2,2,-1,-1)
solnl(x0,objfun=objfun,confun=confun)

####ex2
obj=function(x){
 return((x[1]-1)^2+(x[1]-x[2])^2+(x[2]-x[3])^3+(x[3]-x[4])^4+(x[4]-x[5])^4)
}
#constraint function
con=function(x){
 f=NULL
 f=rbind(f,x[1]+x[2]^2+x[3]^3-2-3*sqrt(2))
 f=rbind(f,x[2]-x[3]^2+x[4]+2-2*sqrt(2))
 f=rbind(f,x[1]*x[5]-2)
 return(list(ceq=f,c=NULL))
}

x0=c(1,1,1,1,1)
solnl(x0,objfun=obj,confun=con)


##########ex3
obj=function(x){
 return((1-x[1])^2+(x[2]-x[1]^2)^2)
}
#constraint function
con=function(x){
 f=NULL
 f=rbind(f,x[1]^2+x[2]^2-1.5)
```

```
 return(list(ceq=NULL,c=f))
}

x0=as.matrix(c(-1.9,2))
obj(x0)
con(x0)
solnl(x0,objfun=obj,confun=con)


##########ex4
objfun=function(x){
 return(x[1]^2+x[2]^2)
}
#constraint function
confun=function(x){
 f=NULL
 f=rbind(f,-x[1] - x[2] + 1)
 f=rbind(f,-x[1]^2 - x[2]^2 + 1)
 f=rbind(f,-9*x[1]^2 - x[2]^2 + 9)
 f=rbind(f,-x[1]^2 + x[2])
 f=rbind(f,-x[2]^2 + x[1])
 return(list(ceq=NULL,c=f))
}

x0=as.matrix(c(3,1))
solnl(x0,objfun=objfun,confun=confun)


##############ex5
rosbkext.f <- function(x){
   n <- length(x)
   sum (100*(x[1:(n-1)]^2 - x[2:n])^2 + (x[1:(n-1)] - 1)^2)
}
n <- 2
set.seed(54321)
p0 <- rnorm(n)
Aeq <- matrix(rep(1, n), nrow=1)
Beq <- 1
lb <- c(rep(-Inf, n-1), 0)
solnl(X=p0,objfun=rosbkext.f, lb=lb, Aeq=Aeq, Beq=Beq)
ub <- rep(1, n)
solnl(X=p0,objfun=rosbkext.f, lb=lb, ub=ub, Aeq=Aeq, Beq=Beq)


##############ex6
nh <- vector("numeric", length = 5)

Nh <- c(6221,11738,4333,22809,5467)
ch <- c(120, 80, 80, 90, 150)

mh.rev <- c(85, 11, 23, 17, 126)
Sh.rev <- c(170.0, 8.8, 23.0, 25.5, 315.0)
```

```
mh.emp <- c(511, 21, 70, 32, 157)
Sh.emp <- c(255.50, 5.25, 35.00, 32.00, 471.00)

ph.rsch <- c(0.8, 0.2, 0.5, 0.3, 0.9)

ph.offsh <- c(0.06, 0.03, 0.03, 0.21, 0.77)

budget = 300000
n.min <- 100
relvar.rev <- function(nh){
 rv <- sum(Nh * (Nh/nh - 1)*Sh.rev^2)
 tot <- sum(Nh * mh.rev)
 rv/tot^2
}

relvar.emp <- function(nh){
 rv <- sum(Nh * (Nh/nh - 1)*Sh.emp^2)
 tot <- sum(Nh * mh.emp)
 rv/tot^2
}

relvar.rsch <- function(nh){
 rv <- sum( Nh * (Nh/nh - 1)*ph.rsch*(1-ph.rsch)*Nh/(Nh-1) )
 tot <- sum(Nh * ph.rsch)
 rv/tot^2
}

relvar.offsh <- function(nh){
 rv <- sum( Nh * (Nh/nh - 1)*ph.offsh*(1-ph.offsh)*Nh/(Nh-1) )
 tot <- sum(Nh * ph.offsh)
 rv/tot^2
}

nlc.constraints <- function(nh){
 h <- rep(NA, 13)
 h[1:length(nh)] <- (Nh + 0.01) - nh
 h[(length(nh)+1) : (2*length(nh)) ] <- (nh + 0.01) - n.min
 h[2*length(nh) + 1] <- 0.05^2 - relvar.emp(nh)
 h[2*length(nh) + 2] <- 0.03^2 - relvar.rsch(nh)
 h[2*length(nh) + 3] <- 0.03^2 - relvar.offsh(nh)
 return(list(ceq=NULL, c=-h))
}

nlc <- function(nh){
 h <- rep(NA, 3)
 h[ 1] <- 0.05^2 - relvar.emp(nh)
 h[ 2] <- 0.03^2 - relvar.rsch(nh)
 h[3] <- 0.03^2 - relvar.offsh(nh)
 return(list(ceq=NULL, c=-h))
}

Aeq <- matrix(ch/budget, nrow=1)
Beq <- 1
```

```
A=rbind(diag(-1,5,5),diag(1,5,5))
B=c(-Nh-0.01,rep(n.min-0.01,5))

solnl(X=rep(100,5),objfun=relvar.rev,confun=nlc.constraints, Aeq=Aeq, Beq=Beq)

solnl(X=rep(100,5),objfun=relvar.rev,confun=nlc, Aeq=Aeq, Beq=Beq, A=-A, B=-B)
```

Index

solnl, 1