Lecture 11: Optimization in R

Cheng Lu

Overview

- General Optimization
  - one dimensional boxed constraint: optimize()
  - multi dimensional without constraint: optim()
  - multi dimensional linear inequality constraint: constrOptim()
- Linear Programming
  - lp() in package "lpSolve"
- Quadratic Programming
  - solve.QP() in package "quadprog"
- Change Problems into Standard Form

General Optimization: optimize()

General optimization problems here are the problems with general objective function. Consider *one dimensional* optimization problem with a box constraint

$$\min_{x \in [l,u]} f(x)$$

where $[l, u] \subseteq \mathbb{R}$. Since x only has one dimension, we can use built-in function **optimize()** to solve the problem. The syntax is given by

```
optimize(f, interval, ..., lower = min(interval), upper = max(interval),
         maximum = FALSE,
         tol = .Machine$double.eps^0.25)
```

For example, if we want to solve the problem

$$\min_{x \in [0,1]} f(x) = (x-a)^2$$

where a is a given constant.

General Optimization: optimize()

When parameter $a = \frac{1}{3}$:

Example

```
> f <- function(x) (x - 1/3)^2 # objective function
> optimize(f, c(0,1))
$minimum
[1] 0.3333333
$objective
[1] 0
> f <- function(x, a) (x - a)^2 # objective function with parameters
> optimize(f, c(0, 1), tol = 0.001, a = 1/3) # set parameters
$minimum
[1] 0.3333333
$objective
[1] 0
```

General Optimization: optim()

For general *multidimensional unconstrained (or only box constrained)* optimization:

$$\min_{x \in \mathbb{R}^n} f(x) \text{ or } \min_{x_i \in [l_i, u_i]} f(x)$$

We can use built-in function **optim()** to solve. The syntax is given by

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

For example, if we want to minimize the Rosenbrock Banana function in $\mathbb{R}^2$:

$$f(x_1, x_2) = 100 * (x_2 - x_1^2)^2 + (1 - x_1)^2$$
$$g(x_1, x_2) = \nabla f(x_1, x_2) = [-400 * x_1 * (x_2 - x_1^2)) - 2 * (1 - x_1), 200 * (x_2 - x_1^2)]$$

General Optimization: optim()

Suppose we want to start at point $(-1.2, 1)$

Example

```
> # Rosenbrock Banana function
> f <- function(x)100*(x[2]-x[1]^2)^2 + (1-x[1])^2
> # gradient
> g <- function(x)c(-400*x[1]*(x[2]-x[1]^2)-2*(1-x[1]), 200*(x[2]-x[1]^2))
> optim(c(-1.2,1), f) # without gradient
$par
[1] 1.000260 1.000506
$value
[1] 8.825241e-08
$counts
function gradient
     195       NA
$convergence
[1] 0
$message
NULL
```

General Optimization: optim()

If we use gradient g, we need to specify the method as "BFGS", "CG" or "L-BFGS-B".

Example

```
> optim(c(-1.2,1), f, g, method = "L-BFGS-B")
$par
[1] 0.9999997 0.9999995
$value
[1] 2.267577e-13
$counts
function gradient
      47       47
$convergence
[1] 0
$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

General Optimization: optim()

If we add box constraint, we need to specify the lower bound and upper bound, and specify the method being **L-BFGS-B**. For example, If we want to add constraint $-2 \leq x_1 \leq 2, 0 \leq x_2 \leq 2$, then *lower* $= (-2, 0)$, and *upper* $= (2, 2)$.

Example

```
> optim(c(-1.2,1), f, lower=c(-2,0), upper=c(2,2), method="L-BFGS-B")
$par
[1] 0.9998502 0.9997006
$value
[1] 2.244198e-08
$counts
function gradient
      71       71
$convergence
[1] 0
$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

General Optimization: constrOptim()

For general optimization with linear inequality constraint:

$$\min_{x} f(x)$$
$$\text{subject to: } Ux \geq c$$

where $U \in \mathbb{R}^{k \times p}$ is a matrix, $c \in \mathbb{R}^k$ is a vector and $x \in \mathbb{R}^p$ is a vector. This is the standard form of built-in function **constrOptim()**. The syntex is given by

```
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)
```

General Optimization: constrOptim()

For example, if we want to solve the following problem:

$$\min_x f(x) = 0.5x_1^2 + 0.5x_2^2 + 0.5x_3^2 - 5x_2$$

$$\text{subject to:} \; -4x_1 - 3x_2 \geq -8$$

$$2x_1 + x_2 \geq 2$$

$$-2x_2 + x_3 \geq 0$$

Then we need to determine the initial point (the "theta" in the syntax), f, ui and ci for this problem. Function f can be define as a function, and the constraint can be written as following

$$f(x) = 0.5 * x^T \begin{bmatrix} 1,0,0 \\ 0,1,0 \\ 0,0,1 \end{bmatrix} x - \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}^T x; U = \begin{bmatrix} -4 & -3 & 0 \\ 2 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}; c = \begin{bmatrix} -8 \\ 2 \\ 0 \end{bmatrix}; Ux \geq c$$

General Optimization: constrOptim()

We can solve the problem without gradient as follows: First we choose a **feasible** starting point "theta", where constraints are satisfied but not necessarily optimal, say $(2, -1, -1)$, then

Example

```
> f <- function(x)0.5*x[1]^2 + 0.5*x[2]^2 + 0.5*x[3]^2 - 5*x[2]
> # f <- function(x)0.5*x%*%diag(3)%*%x - c(0,5,0)%*%x # matrix form
> ui <- matrix(c(-4,-3,0,2,1,0,0,-2,1), 3, 3, byrow =TRUE)
> ci <- c(-8,2,0)
> constrOptim(c(2,-1,-1), f , grad = NULL, ui = ui, ci = ci)$par
[1] 0.4761374 1.0477253 2.0954507
> constrOptim(c(2,-1,-1), f , grad = NULL, ui = ui, ci = ci)$value
[1] -2.380952
```

Here the output of **constrOptim()** is a list, so we use **$par** or **$value** to get the optimal solution and the optimal value respectively.

Linear programming problem is an optimization problem with linear objective function and linear constraint. For example

$$\min_{x\in\mathbb{R}^n} f(x) = \langle c, x\rangle = c^T x$$
$$\text{subject to: } Ax \leq b$$

Here we need to use function **lp()** in package "lpSolve" to solve the problem. The syntax is given by

```
lp (direction = "min", objective.in, const.mat, const.dir, const.rhs,
transpose.constraints = TRUE, int.vec, presolve=0, compute.sens=0,
        binary.vec, all.int=FALSE, all.bin=FALSE, scale = 196, dense.const,
        num.bin.solns=1, use.rw=FALSE)
```

Linear Programming

Suppose we want to find the solution for the following linear programming problem:

$$\max_{x \in \mathbb{R}^3} f(x) = x_1 + 9x_2 + x_3$$

$$\text{subject to: } x_1 + 2x_2 + 3x_3 \leq 9$$

$$3x_1 + 2x_2 + 2x_3 \leq 15$$

According to the syntax of function **lp()**, we need to determine the following input for the function:

- The direction of the optimization problem is "max"
- The objective function can be written as $f(x) = \langle c, x \rangle$, where $c = (1, 9, 1)$
- The constraint can be written as $Ax \leq b$, where

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 2 \end{bmatrix}; b = \begin{bmatrix} 9 \\ 15 \end{bmatrix}$$

- The constraint direction is $\leq$ and $\leq$

Linear Programming

In order to solve the linear programming problem we need to install a package "lpSolve", and use **lp()** function in this package.

Example

```
> install.packages("lpSolve")
> library(lpSolve)
> f.obj <- c(1, 9, 1) # objective
> f.con <- matrix (c(1, 2, 3, 3, 2, 2), nrow=2, byrow=TRUE) # constraint matrix A
> f.dir <- c("<=", "<=") # constraint directions
> f.rhs <- c(9, 15) # right hand side of the constraint b
> lp ("max", f.obj, f.con, f.dir, f.rhs)
Success: the objective function is 40.5
> lp ("max", f.obj, f.con, f.dir, f.rhs)$objval # optimal objective value
[1] 40.5
> lp ("max", f.obj, f.con, f.dir, f.rhs)$solution # optimal solution
[1] 0.0 4.5 0.0
```

Linear Programming

We can also solve integer linear programming problem using the function **lp()**, we can specify which variable is integer using **int.vec =**

Example

```
> # restrict x2 and x3 be integer
> lp ("max", f.obj, f.con, f.dir, f.rhs, int.vec = c(2, 3))$solution
[1] 1 4 0
```

If the objective is

$$f(x) = 9x_2 + x_3 = 0 * x_1 + 9 * x_2 + 1 * x_3$$

then the corresponding vector of objective $c = (0, 9, 1)$. Similar for constraints, for example

$$x_1 \geq 0 \Leftrightarrow 1 * x_1 + 0 * x_2 + 0 * x_3 \geq 0$$

Then we add $(1, 0, 0)$ to the constraint matrix as a row vector, and add ">=" to the direction vector, and add 0 to the right hand side of the constraint.

Linear Programming

Then, for the problem

$$\max_{x\in\mathbb{R}^3} f(x) = 9x_2 + x_3$$
$$\text{subject to: } x_1 + 2x_2 + 3x_3 \leq 9$$
$$3x_1 + 2x_2 + 2x_3 \leq 15$$
$$x_1 \geq 0$$

The constraint matrix A, right hand side vector b, and objective vector c are

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 2 \\ 1 & 0 & 0 \end{bmatrix} ; b = \begin{bmatrix} 9 \\ 15 \\ 0 \end{bmatrix} ; c = \begin{bmatrix} 0 \\ 9 \\ 1 \end{bmatrix}$$

and the constraint direction is c("<=", "<=", ">="), and we can implement it in R.

Linear Programming

Example

```
> f.obj <- c(0 ,9, 1)
> f.con <- matrix(c(1,2,3,3,2,2,1,0,0), nrow=3, byrow=TRUE)
> f.dir <- c("<=", "<=", ">=")
> f.rhs <- c(9, 15, 0)
> lp ("max", f.obj, f.con, f.dir, f.rhs)$solution
[1] 0.0 4.5 0.0
```

Quadratic Programming

Quadratic programming problem is an optimization problem with quadratic objective function and linear constraint. Here we use **solve.QP()** function in package "quadprog" with the standard form

$$\min_x f(x) = \frac{1}{2}x^T Dx - d^T x$$
$$\text{subject to: } A^T x \geq b$$

The syntax is given by

```
solve.QP(Dmat, dvec, Amat, bvec, meq=0, factorized=FALSE)
```

Quadratic Programming

For example, the problem for **constrOptim()**:

$$\min_x f(x) = 0.5x_1^2 + 0.5x_2^2 + 0.5x_3^2 - 5x_2$$

$$\text{subject to:} \quad -4x_1 - 3x_2 \geq -8$$

$$2x_1 + x_2 \geq 2$$

$$-2x_2 + x_3 \geq 0$$

And we need to determine the corresponding D, d, A, and b as the input of the function.

Quadratic Programming

First we write the objective function f in matrix form

$$f(x) = \frac{1}{2}(1x_1^2 + 0x_1x_2 + 0x_1x_3 + 0x_2x_1 + 1x_2^2 + \cdots) - 5x_2 = \frac{1}{2}x^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x - \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}^T x$$

Then we write the constraint in matrix form:

$$\begin{bmatrix} -4 & -3 & 0 \\ 2 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} x \geq \begin{bmatrix} -8 \\ 2 \\ 0 \end{bmatrix}$$

Compare to the standard form of function **solve.QP()**, we have[1]

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; d = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}; A^{\color{red}T} = \begin{bmatrix} -4 & -3 & 0 \\ 2 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}; A = \begin{bmatrix} -4 & 2 & 0 \\ -3 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}; b = \begin{bmatrix} -8 \\ 2 \\ 0 \end{bmatrix}$$

[1] Here each **column** of A corresponds to the coefficient of each constraint. While in other packages, each **row** of constraint matrix corresponds to the coefficient of each constraint. Because there is a transpose for matrix A in standard form of **solve.QP()**.

Quadratic Programming

After writing the problem into matrix form, we can install package "quadprog", then we can use **solve.QP()** function to solve the problem.

Example

```
> install.packages("quadprog")
> library(quadprog)
> D <- matrix(c(1,0,0,0,1,0,0,0,1),3,3) # or D <- diag(1, nrow = 3)
> d <- c(0,5,0)
> A <- matrix(c(-4,-3,0,2,1,0,0,-2,1),3,3) # by column
> b <- c(-8,2,0)
> solve.QP(D, d, A, b)$solution
[1] 0.4761905 1.0476190 2.0952381
> solve.QP(D, d, A, b)$value
[1] -2.380952
```

Change Problems into Standard Form

- Some solvers can only handle minimization problems (e.g. **constOptim()** and **solve.QP()**). So for maximization problem, we need to change it into minimization one:

$$\max_{x \in \mathcal{X}} f(x) = -\min_{x \in \mathcal{X}} \{-f(x)\}$$

  for any feasible set $\mathcal{X}$.

- Some solvers can only handle problems with constraints of direction "$\geq$", if we need to solve the problems with direction "$\leq$", we need to multiply both sides by -1:

$$2x_1 - x_3 \leq 0 \Leftrightarrow -2x_2 + x_3 \geq 0$$

- In **solve.QP()**, if we have constraint with direction "$=$", we need to put the constraints to the first few rows of matrix A, then we set the argument **meq** $=$ the number of the equality constraints.

Change Problems into Standard Form

Suppose we have the following problem

$$\max_x f(x) = -0.5x_1^2 - 0.5x_2^2 - 0.5x_3^2 + 4x_2$$
$$\text{subject to: } 4x_1 + 3x_2 = 8$$
$$2x_2 - x_3 \leq 0$$
$$2x_1 + x_2 = 2$$

First step we need to standardize the problem:

- 1) Change the problem to minimization problem,
- 2) Change the direction of the second constraint,
- 3) Place the equality constraint to the first few rows.

Change Problems into Standard Form

Then the problems becomes

$$-\min_{x}\{-f(x) = 0.5x_1^2 + 0.5x_2^2 + 0.5x_3^2 - 4x_2\}$$

$$\text{subject to: } 4x_1 + 3x_2 = 8$$

$$2x_1 + x_2 = 2$$

$$-2x_2 + x_3 \geq 0$$

Compare to the matrix form, we have

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; d = \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix}; A = \begin{bmatrix} 4 & 2 & 0 \\ 3 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}; b = \begin{bmatrix} 8 \\ 2 \\ 0 \end{bmatrix}; \mathbf{meq = 2}$$

Change Problems into Standard Form

We can first solve the minimization problem, then multiply the objective value by -1.

Example

```
> D <- diag(c(1,1,1))
> d <- c(0,4,0)
> A <- matrix(c(4,3,0,2,1,0,0,-2,1),3,3)
> b <- c(8,2,0)
> solve.QP(D, d, A, b, meq = 2)$solution # optimal solution x
[1] -1  4  8
> solve.QP(D, d, A, b, meq = 2)$value # minimum value of -f(x)
[1] 24.5
> -solve.QP(D, d, A, b, meq = 2)$value # maximum value of f(x)
[1] -24.5
```

The optimal solution to the maximization problem is $(-1, 4, 8)$ with the optimal value -24.5.