Tsinghua University
Teammates(alphabetically): 付郁婷，数52，2015012085
李映雪，数52，2015012066；吕泽群，数51，2015012032
宋歌，数52，2015080086；孙凌宇，数52，2015012071

Project 2: ADMM for Linear Programming

Due June 8, 2018

Consider solving the linear program

$$
\begin{aligned}
min \quad & c^T x \\
s.t. \quad & Ax = b \\
& x \geqslant 0
\end{aligned}
$$

or its dual

$$
\begin{aligned}
min \quad & b^T y \\
s.t. \quad & A^T y + s = c \\
& s \geqslant 0
\end{aligned}
$$

The Augmented Lagrangian function would be

$$L^p(x, y) = c^T x - y^T(Ax - b) + \frac{\beta}{2}||Ax - b||^2$$

where β is a positive parameter, for the primal; and

$$L^d(y, s, x) = -b^T y - x^T(A^T y + s - c) + \frac{\beta}{2}||A^T y + s - c||^2$$

for the dual.

1. ADMM for the Primal

1) Theoretical Analysis

We reformulate the LP problem as

$$
\begin{aligned}
min \quad & c^T x_1 \\
s.t. \quad & Ax_1 = b \\
& x_1 - x_2 = 0 \\
& x_2 \geqslant 0
\end{aligned}
$$

and consider the split augmented Lagrangian function:

$$L^p(x_1, x_2, y, s) = c^T x_1 - y^T(Ax_1 - b) - s^T(x_1 - x_2) + \frac{\beta}{2}(||Ax_1 - b||^2 + ||x_1 - x_2||^2)$$

Then the **Alternating Direction Method with Multipliers (ADMM)** would be:

starting from any $x_1^0, x_2^0 \geqslant 0$, and multiplier (y^0, s^0), do the iterative update:

- Update variable x_1:

$$x_1^{k+1} = \arg\min_{x_1} L^p(x_1, x_2^k, y^k, s^k)$$

$$\begin{aligned}
\nabla_{x_1} L^p(x_1, x_2^k, y^k, s^k) &= (\beta A^T A + \beta)x_1 + c - A^T y^k - s^k - \beta A^T b - \beta x_2^k \\
\Delta_{x_1} L^p(x_1, x_2^k, y^k, s^k) &= A^T A \beta^T + \beta^T \geqslant 0 \qquad \text{since } \beta > 0
\end{aligned}$$

Thus, for fixed x_2^k, y^k, s^k, $L^p(x_1, x_2^k, y^k, s^k)$ is a convex function with respect to x_1. Furthermore, this subproblem is an unconstrained convex problem, so we know that x_1^{k+1} is an optimal solution if and only if $\nabla_{x_1} L^p(x_1^{k+1}, x_2^k, y^k, s^k) = 0$. We assume that A has full rank, then $A^T A + I$ is positive definite and we obtain:

$$x_1^{k+1} = \frac{1}{\beta}(A^T A + I)^{-1}(\beta x_2^k + \beta A^T b + s^k + A^T y^k - c)$$

- Update variable x_2:

$$x_2^{k+1} = \arg\min_{x_2 \geqslant 0} L^p(x_1^{k+1}, x_2, y^k, s^k)$$

$$\begin{aligned}
\nabla_{x_2} L^p(x_1^{k+1}, x_2, y^k, s^k) &= \beta x_2 + s^k - \beta x_1^{k+1} \\
\Delta_{x_2} L^p(x_1^{k+1}, x_2, y^k, s^k) &= \beta > 0
\end{aligned}$$

Thus, for fixed x_1^{k+1}, y^k, s^k, $L^p(x_1^{k+1}, x_2, y^k, s^k)$ is a convex function with respect to x_2. Furthermore, this subproblem is a linearly constrained convex problem, so we know that the KKT point is an optimal point. We can write this subproblem with respect to x_2 as

$$\begin{aligned}
min \quad & \frac{\beta}{2}||x_2 - x_1^{k+1}||^2 + s^{kT} x_2 \\
s.t. \quad & x_2 \geqslant 0
\end{aligned}$$

And the KKT conditions for this problem are

$$\begin{aligned}
& s^k + \beta x_2^{k+1} - \beta x_1^{k+1} - \lambda = 0 \\
& x_2^{k+1} \geqslant 0 \\
& \lambda \geqslant 0 \\
& \lambda^T x_2^{k+1} = 0
\end{aligned}$$

We know consider the following two situations:

$$\begin{aligned}
\lambda = 0 \Rightarrow x_2^{k+1} = x_1^{k+1} - \frac{1}{\beta}s^k \Rightarrow L^p = s^{kT} x_1 k + 1 - \frac{1}{2\beta}||s^k||^2 \\
\lambda > 0 \Rightarrow x_2^{k+1} = 0 \Rightarrow \lambda = s^k - \beta x_1^{k+1} \Rightarrow L^p = \frac{\beta}{2}||x_1^{k+1}||^2
\end{aligned}$$

Since

$$\frac{\beta}{2}||x_1^{k+1}||^2 - s^{kT} x_1 k + 1 + \frac{1}{2\beta}||s^k||^2 = \frac{\beta}{2}||x_1^{k+1} - \frac{1}{\beta s^k}||^2 \geqslant 0$$

We obtain

$$x_2^{k+1} = \max\{0, x_1^{k+1} - \frac{1}{\beta}s^k\}$$

- Update multipliers y and s:

$$y^{k+1} = y^k - \beta(Ax_1^{k+1} - b)$$
$$s^{k+1} = s^k - \beta(x_1^{k+1} - x_2^{k+1})$$

2) Algorithm Implementation

- ADMM function for LP

```
#define function Max
Max <- function(X){
  n <- length(X)
  for(i in 1:n){
    if(X[i] >= 0){
      X[i] <- X[i]
    }
    else{
      X[i] <- 0
    }
  }
  return(X)
}

#admm solver for LP
admm_lp <- function(A, b, c, beta){
  #global
  m <- nrow(A)
  n <- ncol(A)

  eps <- 0.0001
  maxiter <- 100

  #initialization
  objval <- vector()
  x_1 <- x_2 <- s <- rep(0,n)
  dim(x_1) <- dim(x_2) <- dim(s) <- c(n,1)
  y <- rep(0,m)
  dim(y) <- c(m,1)


  #ADMM
  for(k in 1:maxiter){
    #update x_1
    x_1 <- (1/beta) * solve(t(A)%*%A + diag(n), beta*x_2 + beta*t(A)%*%b + s + t(A)%*%y - c)
    #update x_2
    x_2 <- Max(x_1 - (1/beta) * s)
    #update y
    y <- y - beta * (A%*%x_1 - b)
    #update s
    s <- s - beta * (x_1 - x_2)

    #store objective value
    objval[k] <- t(c)%*%x_1

    #show iteration
    cat(k,objval[k],fill=T)

    #termination
    if(norm(as.matrix(x_1 - x_2)) < eps & norm(as.matrix(A%*%x_1 - b)) < eps){
      break;
    }
  }

  #plot
  K <- length(objval)
  mydata <- data.frame(iter = 1:K, objective = objval)
  ggplot(mydata, aes(x = iter, y = objective)) + geom_line(size = 1, col = 'red')
}
```

- Example

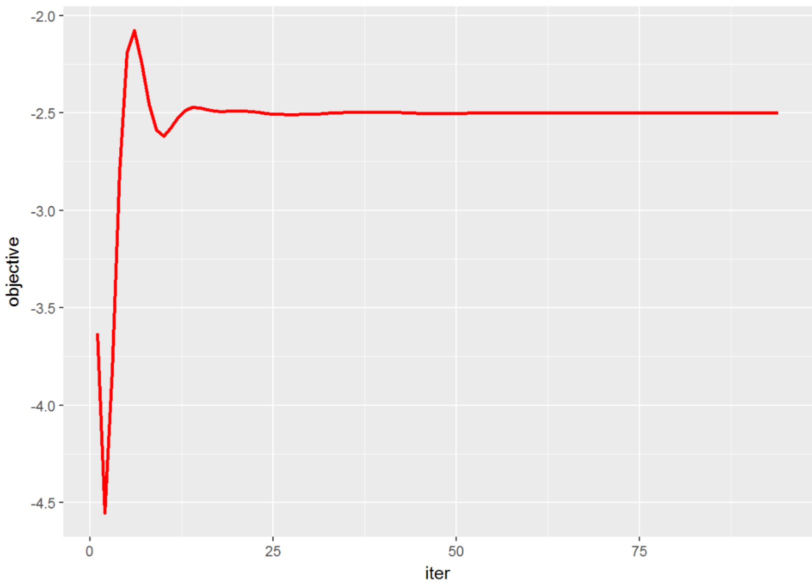
```r
#eg1-lp
A <- matrix(nrow = 3, ncol = 5)
b <- matrix(nrow = 3, ncol = 1)
c <- matrix(nrow = 5, ncol = 1)

A[1,] <- c(1,0,1,0,0)
A[2,] <- c(0,1,0,1,0)
A[3,] <- c(1,1,0,0,1)

b[,1] <- c(1,1,1.5)
c[,1] <- c(-1,-2,0,0,0)

admm_lp(A,b,c,1)
```

```
## 80 -2.499433
## 81 -2.499556
## 82 -2.499708
## 83 -2.499874
## 84 -2.500036
## 85 -2.50018
## 86 -2.500293
## 87 -2.500367
## 88 -2.5004
## 89 -2.500392
## 90 -2.500347
## 91 -2.500274
## 92 -2.500182
## 93 -2.500082
## 94 -2.499983
```

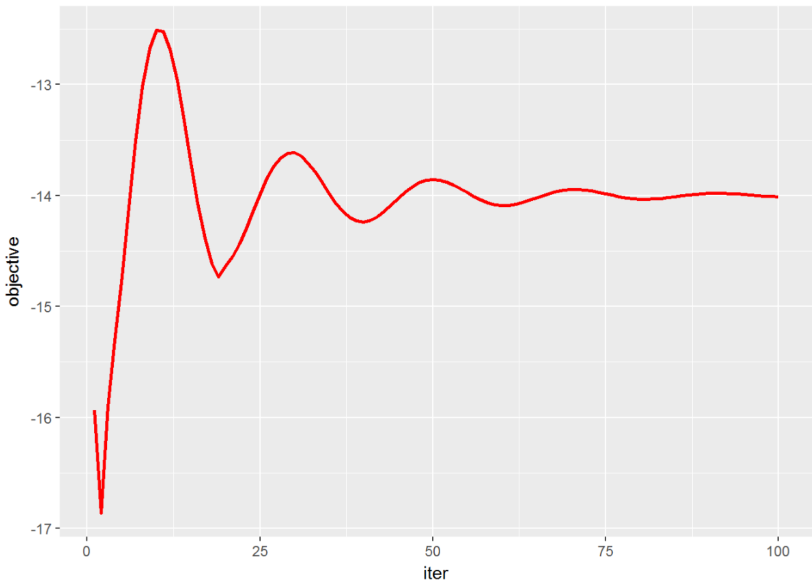


```r
#eg2-lp
A <- matrix(nrow = 3, ncol = 5)
b <- matrix(nrow = 3, ncol = 1)
c <- matrix(nrow = 5, ncol = 1)

A <- matrix(c(1,4,0,2,0,4,1,0,0,0,1,0,0,0,1), nrow = 3)
b[,1] <- c(8,16,12)
c[,1] <- c(-2,-3,0,0,0)

admm_lp(A,b,c,1)
```

```
## 80 -14.03253
## 81 -14.03355
## 82 -14.03143
## 83 -14.02664
## 84 -14.01986
## 85 -14.01188
## 86 -14.00356
## 87 -13.99566
## 88 -13.98888
## 89 -13.98373
## 90 -13.98053
## 91 -13.9794
## 92 -13.98024
## 93 -13.9828
## 94 -13.9867
## 95 -13.99145
## 96 -13.99655
## 97 -14.0015
## 98 -14.00586
## 99 -14.00929
## 100 -14.01156
```

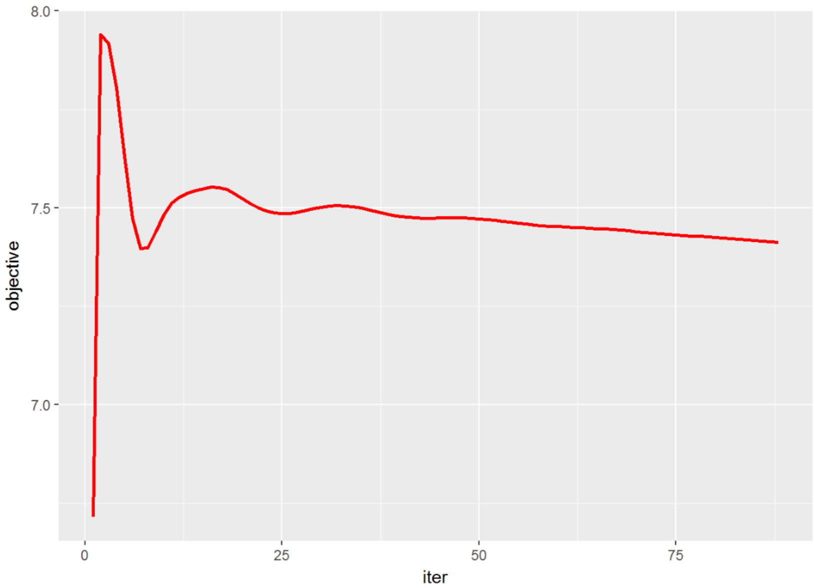


```r
#eg3-1d
A <- matrix(nrow = 4, ncol = 5)
b <- matrix(nrow = 4, ncol = 1)
c <- matrix(nrow = 5, ncol = 1)
x_0 <- matrix(nrow = 5, ncol = 1)

set.seed(2015080086)
A[,] <- abs(rnorm(4*5,0,1))
x_0[,1] <- abs(rnorm(5,0,1))
b[,1] <- A%*%x_0
c[,1] <- rnorm(5,0,1) + 0.5

admm_lp(A,b,c,100)
```

```
## 70 7.439679
## 71 7.437918
## 72 7.436184
## 73 7.43451
## 74 7.432914
## 75 7.431403
## 76 7.429972
## 77 7.428603
## 78 7.427275
## 79 7.425965
## 80 7.424647
## 81 7.423303
## 82 7.421919
## 83 7.420489
## 84 7.419014
## 85 7.4175
## 86 7.415959
## 87 7.414404
## 88 7.412847
```



2. ADMM for the Dual

1) Theoretical Analysis

The augmented Lagrangian function for the dual problem can be written as:

$$L^d(y, s, x) = -b^T y - x^T(A^T y + s - c) + \frac{\beta}{2}||A^T y + s - c||^2$$

- Update variable y:

$$y^{k+1} = \arg\min_{y} L^d(y, s^k, x^k)$$

$$\begin{aligned} \nabla_y L^d &= -b - Ax^k + \beta AA^T y + \beta A(s^k - c) \\ \Delta_y L^d &= \beta AA^T \geqslant 0 \end{aligned}$$

Thus, for fixed s^k, x^k, $L^d(y, s^k, x^k)$ is a convex function with respect to y. Furthermore, this subproblem is an unconstrained convex problem, so we know that y^{k+1} is an optimal solution if and only if $\nabla_y L^d(y^{k+1}, s^k, x^k) = 0$. We assume that A has full rank, then $A^T A$ is positive definite and we obtain:

$$y^{k+1} = \frac{1}{\beta}(AA^T)^{-1}(b + Ax^k + \beta A(c - s^k))$$

- Update slack variable s:

$$s^{k+1} = \arg\min_{s \geq 0} L^d(y^{k+1}, s, x^k)$$

$$\begin{aligned} \nabla_s L^d &= -x^k + \frac{\beta}{2}(2s + 2(A^T y^{k+1} - c)) - x^k + \beta s + \beta(A^T y^{k+1} - c) \\ \Delta_s L^d &= 2\beta > 0 \end{aligned}$$

Thus, for fixed y^{k+1}, x^k, $L^d(y^{k+1}, s, x^k)$ is a convex function with respect to s. Furthermore, this subproblem is a linearly constrained convex problem, so we know that the KKT point is an optimal point. Similar to the process in primal problem, we obtain:

$$s^{k+1} = \max\{0, \frac{1}{\beta}x + c - A^T y^{k+1}\}$$

- Update multipliers x:

$$x^{k+1} = x^k - \beta(A^T y^{k+1} + s^{k+1} - c)$$

2) Algorithm Implementation

- ADMM function for LD

```
admm_ld <- function(A, b, c, ep, beta){
ep=0.0001
beta=10

Y=matrix(0, nrow=nrow(A), ncol=100)
X=matrix(0, nrow=ncol(A), ncol=100)
S=matrix(0, nrow=ncol(A), ncol=100)

X[, 1]=rep(0, ncol(A))
Y[, 1]=rep(0, nrow(A))
S[, 1]=rep(1, ncol(A))

objval=vector()
index=0
a=FALSE
for (i in 1:99)
{
  Y[, i+1]=(1/beta)*solve(A%*%t(A))%*%(b+A%*%X[, i]+beta*A%*%(c-S[, i]))
  vec=(1/beta)*X[, i]+c-t(A)%*%Y[, i+1]
  for(k in 1:length(vec))
  {
    if(vec[k]<0)
    {
      S[k, i+1]=0
    }
    else
    {
      S[k, i+1]=vec[k]
    }
  }
  #if(min(0, vec)<0)
  #{
  #  S[, i+1]=0
  #}
  #else
  #{
  #  S[, i+1]=vec
  #}
  X[, i+1]=X[, i]-beta*(t(A)%*%Y[, i+1]+S[, i+1]-c)
  objval[i]=Y[, i]%*%b
  if(norm(as.matrix((Y[, i+1]-Y[, i])))<ep)
  {
    a=TRUE
    index=i
  }
  if(a==TRUE)
    break
}


print(index)
print(Y[, index])
print(objval[index])

K <- index
mydata <- data.frame(iter = 1:K, objective = objval[1:K])
ggplot(mydata, aes(x = iter, y = objective)) + geom_line(size = 1, col = 'red')

}
```

(The algorithms for LP and LD are from two different teammates, so there are some slight differences.)

- Example

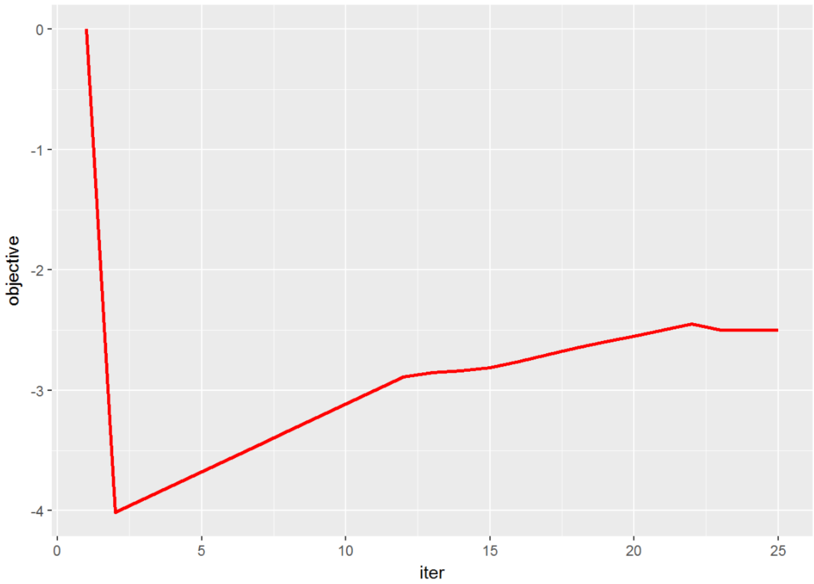
```r
#eg1-ld
A <- matrix(nrow = 3, ncol = 5)
b <- matrix(nrow = 3, ncol = 1)
c <- matrix(nrow = 5, ncol = 1)

A[1,] <- c(1,0,1,0,0)
A[2,] <- c(0,1,0,1,0)
A[3,] <- c(1,1,0,0,1)

b[,1] <- c(1,1,1.5)
c[,1] <- c(-1,-2,0,0,0)

admm_ld(A,b,c,0.0001,10)
```

```
## [1] 25
## [1] -5.920306e-05 -1.000102e+00 -9.998816e-01
## [1] -2.499984
```

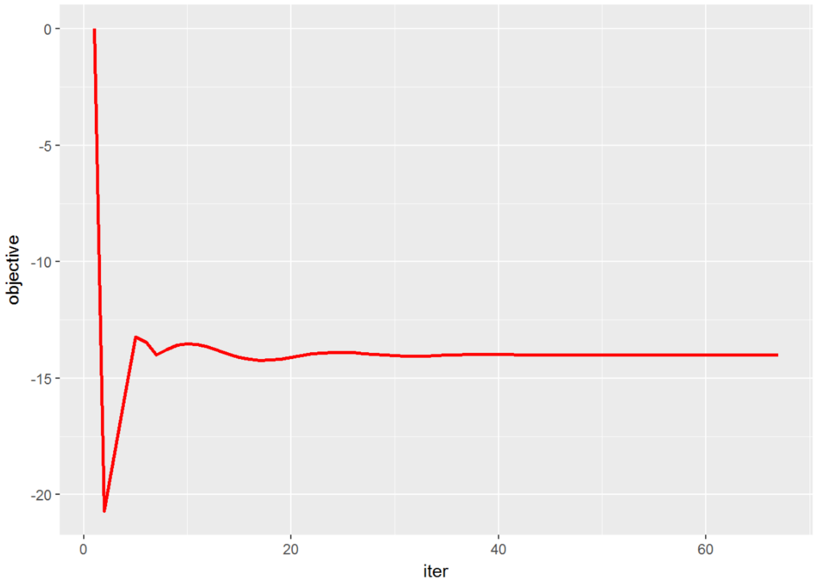


```r
#eg2-1d
A <- matrix(nrow = 3, ncol = 5)
b <- matrix(nrow = 3, ncol = 1)
c <- matrix(nrow = 5, ncol = 1)

A <- matrix(c(1,4,0,2,0,4,1,0,0,0,1,0,0,0,1), nrow = 3)
b[,1] <- c(8,16,12)
c[,1] <- c(-2,-3,0,0,0)

admm_1d(A,b,c,0.0001,10)
```

```
## [1] 67
## [1] -1.5009239417 -0.1247657754  0.0004347961
## [1] -13.99843
```

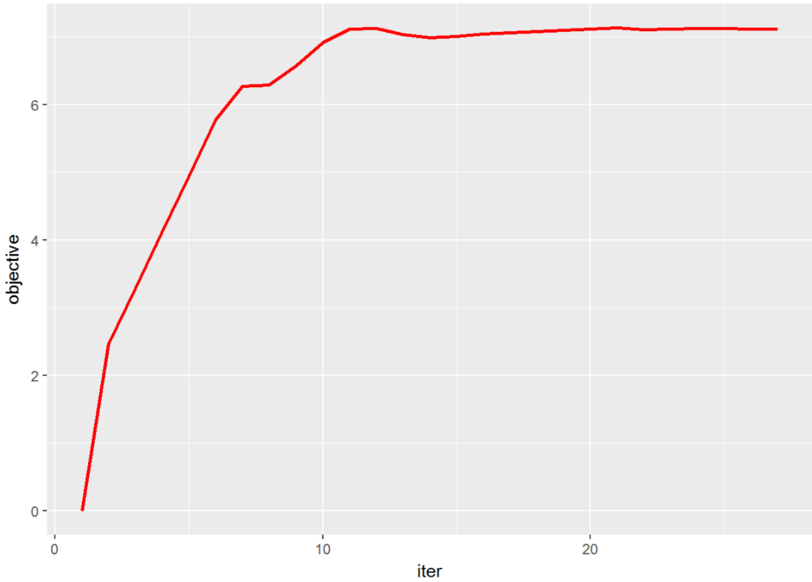


```r
#eg3-ld
A <- matrix(nrow = 4, ncol = 5)
b <- matrix(nrow = 4, ncol = 1)
c <- matrix(nrow = 5, ncol = 1)
x_0 <- matrix(nrow = 5, ncol = 1)

set.seed(2015080086)
A[,] <- abs(rnorm(4*5,0,1))
x_0[,1] <- abs(rnorm(5,0,1))
b[,1] <- A%*%x_0
c[,1] <- rnorm(5,0,1) + 0.5

admm_ld(A,b,c,0.0001,100)
```

```
## [1] 27
## [1]  0.2944572  1.2514484  1.0566257 -0.4196678
## [1] 7.118049
```



可见收敛速度较快，且相应例子的LP和LD问题所得最优目标函数值十分接近。