ADMM for LASSO

LASSO regression problem is to solve $\hat{\beta} = argmin_\beta\{\frac{1}{2}\|y - X\beta\|_2^2 + \gamma|\beta|_1\}$, i.e. to find $\hat{\beta}$ that minimize the objective function:

$$\underset{\beta}{minimize}\{\tfrac{1}{2}\|y - X\beta\|_2^2 + \gamma|z|_1\}$$

It is common to use the constraint that the $L1$ norm of the parameter vector, is no greater than a given value. In this instance unfortunately it is non-trivial, and not terribly efficient, to apply soft thresholding to this problem as each element of β cannot be treated independently. Instead, however, we can apply Alternating Direction Method of Multipliers.

The general form of ADMM in LASSO can be written as follows. It allows us to employ a dummy variable into the objective such that:

$$\begin{aligned}&\underset{\beta,z}{argmin}\tfrac{1}{2}\|y - X\beta\|_2^2 + \gamma|z|_1 + \tfrac{\rho}{2}\|\beta - z\|_2^2\\ &\text{subject to } \beta - z = 0\end{aligned}$$

This objective seems to complicate things. However, we can form the *Augmented Lagrangian* of the above objective which becomes:

$$L(\beta, z, \lambda) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \gamma|z|_1 + \tfrac{\rho}{2}\|\beta - z\|_2^2 + \lambda(\beta - z)$$

The ADMM algorithm consists of the iterations:

Step 1. Set initial z^0, λ^0;

Step 2. (1) $\beta^{t+1} = \underset{\beta}{argmin}L(\beta, z^t, \lambda^t)$;

(2) $z^{t+1} = \underset{z}{argmin}L(\beta^{t+1}, z, \lambda^t)$;

(3) $\lambda^{t+1} = \lambda^t + \rho(\beta^{t+1} - z^{t+1})$;

Step 3. Repeat Step 2 until converges.

which allows us to break our original problem into a sequence of 3 sub-problems.

Sub-problem 1. If we are minimizing $L()$ w.r.t. only β then the $L1$ penalty $|z|_1$ disappears from the objective, making it a very efficient and simple least-squares regression problem. β can be solved by FOC (First Order Condition):

$$\frac{\partial}{\partial\beta}L = -(y - X\beta)^T X + \rho(\beta - z) + \lambda = 0$$

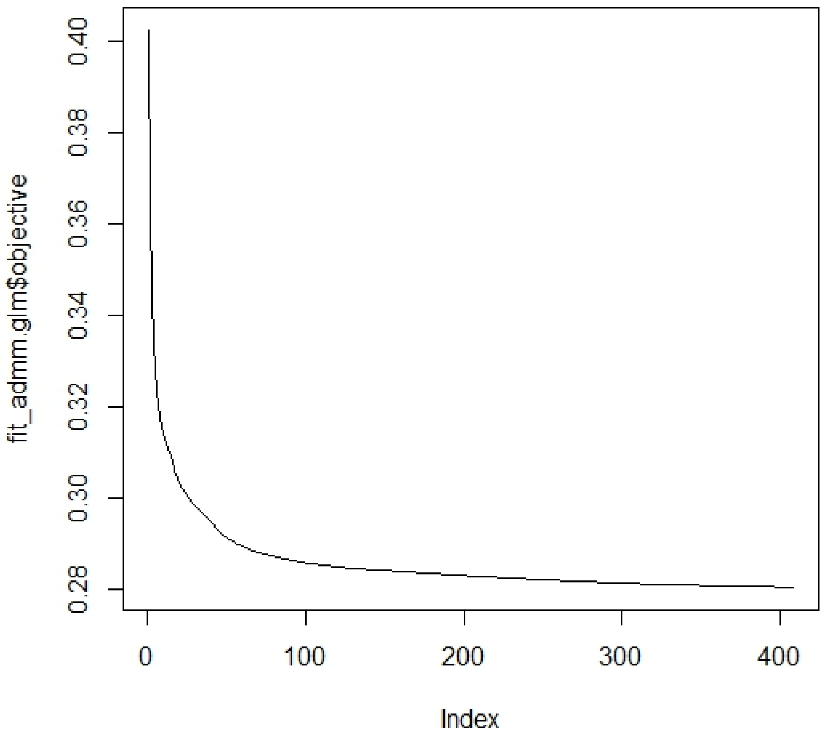
$$\beta = (X^T X + \rho I)^{-1}(X^T y + \rho z - \lambda)$$

Sub-problem 2. If we are minimizing $L()$ w.r.t. only z then the term $\|y - X\beta\|_2^2$ disappears allowing for z to be solved independently across each element. This now allows us to use soft-thresholding efficiently.

$$z = sign(\beta)(|\beta| - \frac{\lambda}{\rho})_+$$

The current estimates of β and z are then combined in step 3 of the ADMM to update our current estimate of the Lagrangian multiplier matrix λ.

Sub-problem 3. The penalty parameter ρ that was introduced into our objective plays a special role here, as it allows us to employ an imperfect estimate of λ when solving for both β and z.



Code:

```
Diabetes = read.csv( "C:/Users/Yuxin/Dropbox/Courses/2016 Fall/Stat Model for Big Data/E

y=scale(Diabetes$Y)
N=length(y)
p=ncol(Diabetes)-1
x=scale(Diabetes[, -1])

objective = function (obj, x, y, beta, z, rho, gamma){
obj.t=0.5/ N * sum( (y-x%*%beta[,t+1] )^2 ) + gamma* sum(abs(z[,t+1])) + rho/2 * sum( (b
obj=c(obj, obj.t)

return(obj)
}

betat=function( beta, x, y, z, rho, lambda  ){
beta.t= solve(1/N * t(x)%*%x + (rho)* diag(1,p) ) %*% ( 1/N *t(x) %*% y + (rho) * z[,t]
beta=cbind(beta, beta.t)

return( beta )
}

prox=function(beta, gamma, rho ){

prox=sign(beta) * pmax( ( abs(beta) -  gamma/ (rho) ), 0)

return(prox)
}

zt=function( z, beta, lambda, rho ){
```

```
z.t=prox( beta[,t+1] , lambda[t] , rho )
z=cbind(z,z.t)

return(z)
}

lambdat=function( lambda, rho, beta, z ){
lambda.t=lambda[t]+(rho)*(beta[,t+1] -z[,t+1] )
lambda=c(lambda, lambda.t)

return(lambda)
}

ADMM.lasso=function( p, y, x, beta1, lambda1, gamma, rho, maxiter, tol ){

beta=beta1
z=beta1
lambda=lambda1

t=1

obj = 0.5/ N * sum( (y-x%*%beta[,t])^2 ) + gamma* sum(abs(z[,t])) + rho/2 * sum( (beta[,

while (t < maxiter  ){

beta=betat( beta=beta, x=x, y=y, z=z, rho=rho, lambda=lambda )
z=zt( z=z, beta=beta, lambda=lambda, rho=rho )
lambda=lambdat( lambda=lambda, rho=rho, beta=beta, z=z )
obj=objective(obj=obj, x=x, y=y, beta=beta, z=z, rho=rho, gamma=gamma)

# if (abs(obj[t+1] - obj[t] ) < tol){
# break
# }


t=t+1

}


return( list( beta=beta, z=z, lambda=lambda, t=t, obj=obj ) )

}



beta1=matrix(0.01, nrow=p, ncol=1)

lambda1=0.1

gamma=0.002

rho=1

maxiter=500

tol=1e-4


fit.ADMM.lasso=ADMM.lasso( p=p, y=y, x=x, beta1=beta1, lambda1=lambda1, gamma=gamma, rho
```