(A)

(1) Negative log-likelihood of $N(\theta,1)$ is

$-log(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-\frac{(y-\theta)^2}{2}}) \propto -log(e^{-\frac{(y-\theta)^2}{2}}) = \frac{1}{2}(y-\theta)^2$

(2) $S_\lambda(y) = argmin_\theta \frac{1}{2}(y-\theta)^2 + \lambda|\theta|$

i) For $\theta \geq 0$,

$S_\lambda(y) = \frac{1}{2}y^2 - y\theta + \frac{1}{2}\theta^2 + \lambda\theta = \frac{1}{2}\theta^2 + (\lambda - y)\theta + \frac{1}{2}y^2$

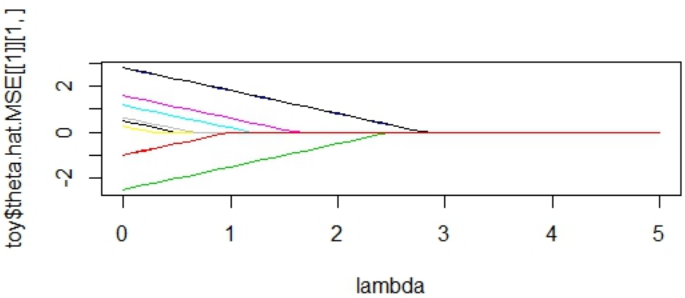
$\frac{d}{d\theta}S_\lambda(y) = \theta + (\lambda - y)$, $\theta = y - \lambda = sign(y)(|y| - \lambda)_+$

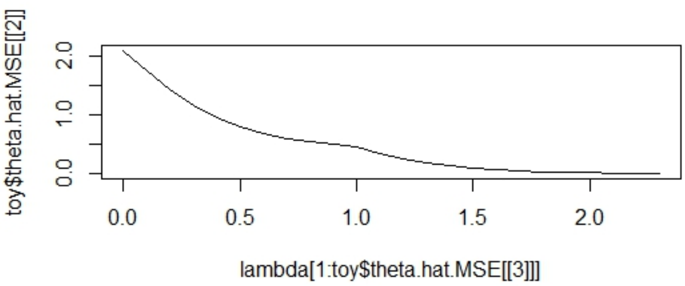
ii) For $\theta < 0$,

$S_\lambda(y) = \frac{1}{2}y^2 - y\theta + \frac{1}{2}\theta^2 - \lambda\theta = \frac{1}{2}\theta^2 - (\lambda + y)\theta + \frac{1}{2}y^2$

$\frac{d}{d\theta}S_\lambda(y) = \theta - (\lambda + y)$, $\theta = y + \lambda = sign(y)(-y+\lambda)_+ = sign(y)(|y| - \lambda)_+$

Following are 2 graphs for the convergence of $\hat{\theta}$ and the MSE





Code for A

```
toy=function(l.theta, sparse.rate, sigma){
z.theta=generate.z.theta(l.theta, sparse.rate, sigma )
theta.hat.MSE=theta.hat.MSE( z.theta=z.theta, lambda )
```

```
return( list(z.theta=z.theta, theta.hat.MSE=theta.hat.MSE ) )
}

generate.z.theta=function(l.theta, sparse.rate, sigma ){
theta=rep(0, l.theta)

sparse.index=sample.int(l.theta, l.theta * sparse.rate)
theta[sparse.index]=0
theta[-sparse.index]=runif(1,0,1)

sigma=diag(rep(sigma,l.theta))
z=mvrnorm(1,theta,sigma)

return( list(theta=theta, z=z) )
}

theta.hat.MSE=function( z.theta, lambda ){

z=z.theta$z
theta=z.theta$theta

l.lambda=length(lambda)

theta.hat=matrix(0, nrow=l.theta, ncol=l.lambda)
S_lambda=rep(0,l.lambda)
MSE0=c()

for (i in 1:l.lambda){
theta.hat[,i]=sign(z)* (abs(z)-lambda[i]) * as.numeric(sign(abs(z)-lambda[i])>=0)
MSE0=c(MSE0, 1/l.theta*sum(theta.hat[,i] - theta)^2 )
}
```

```
which.min=which.min(MSE0)
MSE=MSE0[1:which.min]

return( list(theta.hat, MSE, which.min)  )
}



l.theta=10
sparse.rate=0.8
sigma=3

lambda=seq(from=0, to=5, by=0.1)

toy=toy(l.theta, sparse.rate, sigma)
toy

#### Plot ####
par(mfrow = c(2,1))

####  theta Plot ####
plot(lambda, toy$theta.hat.MSE[[1]][1,], type="l", ylim=c(min(toy$theta.hat.MSE[[1]]), m
for(i in 2:l.theta){
lines(lambda, toy$theta.hat.MSE[[1]][i,], col=i)

}
####  MSE Plot ####
plot(lambda[1:toy$theta.hat.MSE[[3]]],toy$theta.hat.MSE[[2]], type="l")
```

(B)

$S_{\lambda\sigma_i^2}(y) = argmin_{\theta_i} \frac{1}{2\sigma_i^2}(y-\theta_i)^2 + \lambda|\theta_i|$

i) For $\theta_i \geq 0$,

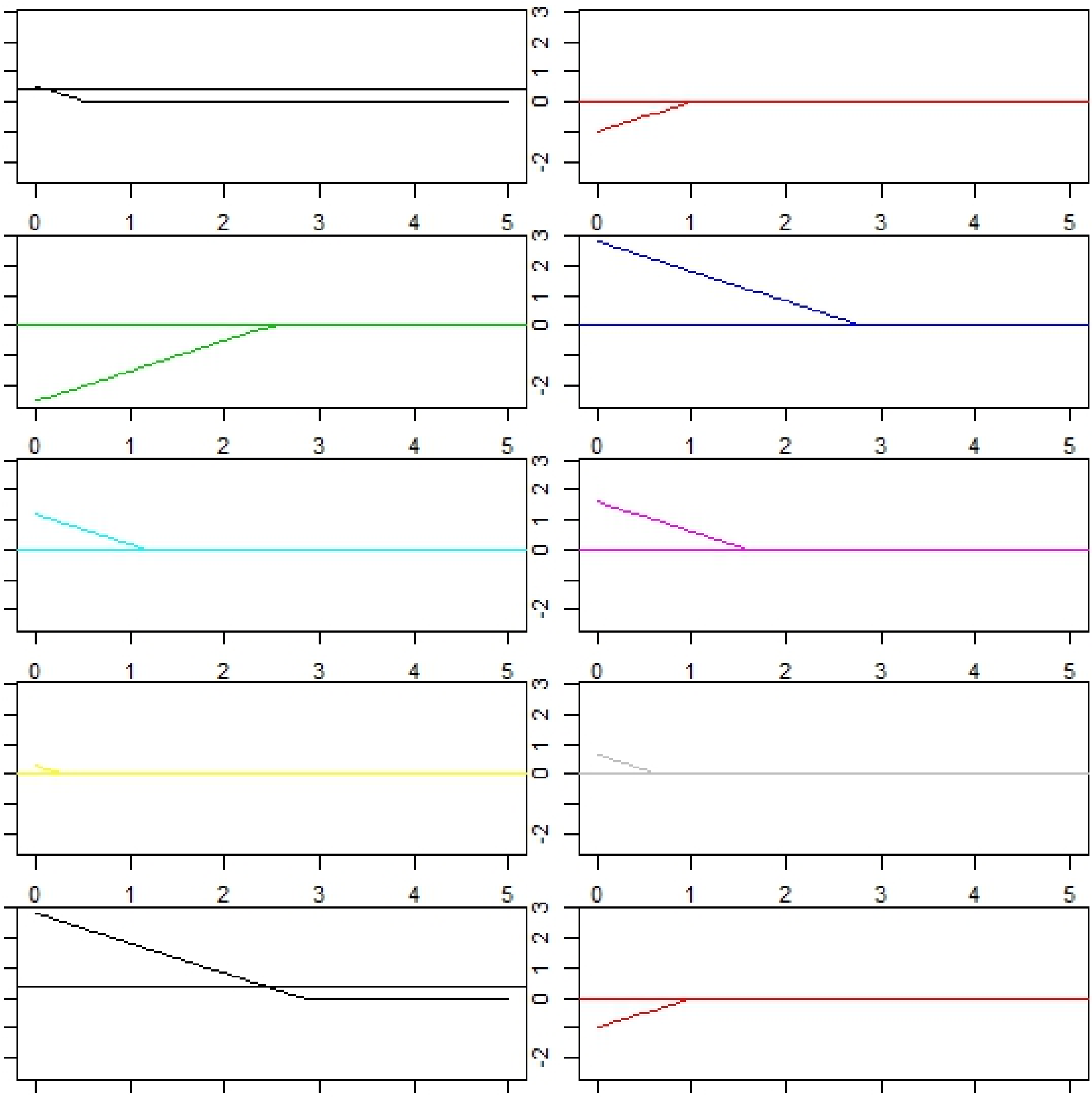
$S_{\lambda\sigma_i^2}(y) = \frac{1}{2}\frac{(y^2 - y\theta_i + \theta_i^2)}{\sigma_i^2} + \lambda\theta_i = \frac{1}{2\sigma_i^2}(\theta^2 + (\lambda\sigma_i^2 - y)\theta + y^2)$

$\theta_i = y - \lambda\sigma_i^2 = sign(y)(|y| - \lambda\sigma_i^2)_+$

ii) For $\theta_i < 0$,

By similar induction, $\theta_i = y - \lambda\sigma_i^2 = sign(y)(|y| - \lambda\sigma_i^2)_+$

Following are the plots for $\hat{\theta}$ vs. θ

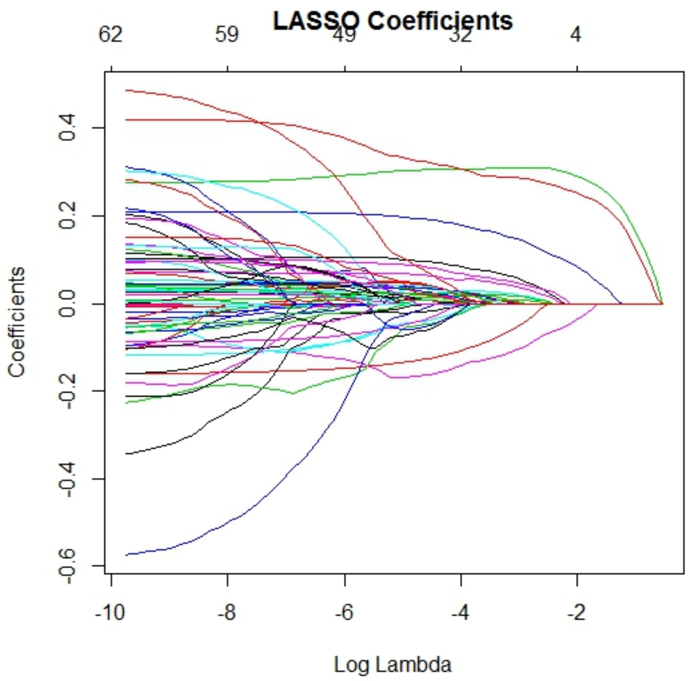


Code for Plotting

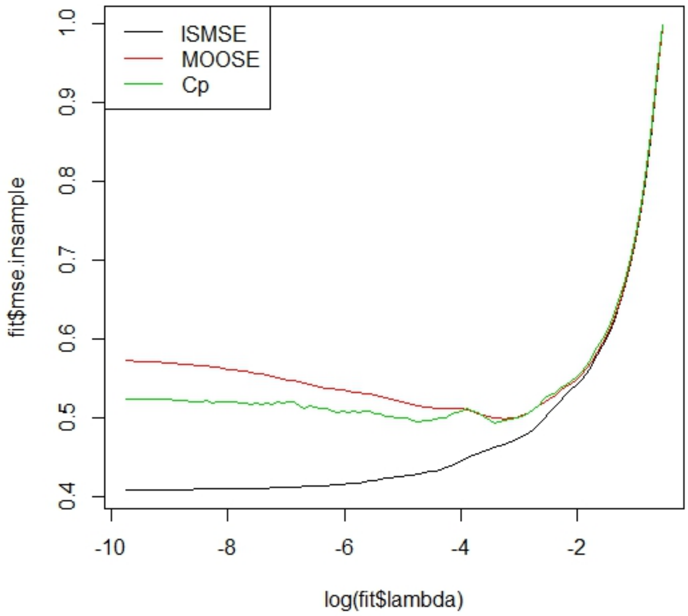
```
#### theta.hat vs. theta ####
par(mar=c(1,1,1,1))
par(mfrow = c(5,2))
for(i in 1:l.theta){
plot(lambda, toy$theta.hat.MSE[[1]][i,], type="l", ylim=c(min(toy$theta.hat.MSE[[1]]), m
abline(a=toy$z.theta[[1]][i], b=0, col=i)
}
```

(C)

The following is the plot for Coefficients.



Compare three methods.




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

y=scale(Diabetes$Y)
P=ncol(Diabetes)-1
x=scale(Diabetes[, 2:P])

fit.lasso.mse=function( x, y ){

fit.lasso = glmnet(x,y, family = 'gaussian')

lambda = fit.lasso$lambda
beta = fit.lasso$beta
l.lambda = length(lambda)

mse.insample = rep(0,l.lambda)
for (i in 1:l.lambda){
    mse.insample[i] = sum((y - x %*% beta[,i])^2) / nrow(x)
}
```

```
return(list(fit.lasso=fit.lasso, lambda=lambda, beta=beta, mse.insample=mse.insample ) )
}

fit=fit.lasso.mse( x, y )
Df=fit$fit.lasso$df

#### Cross Validation ####

lambda = fit$lambda

crossvalidation = function(y, x, split = 10, lambda){

folds=createFolds(t(y), k = split, list = TRUE, returnTrain = FALSE)

    pred_error = matrix(,nrow = split, ncol = length(lambda))

    #Perform 10 split cross validation
    for(i in 1:split){
        x.train=x[-folds[[i]],]
y.train=y[-folds[[i]]]
x.test=x[folds[[i]],]
y.test=y[folds[[i]]]
        fit.train = glmnet(x = x.train, y = y.train, family = 'gaussian',lambda = lambda
        pred = predict(fit.train, newx = x.test, s = lambda)
prederr = pred - y[folds[[i]]]
        pred_error[i,] = colMeans(prederr^2)
    }

return(list(lambda = lambda, MOOSE = colMeans(pred_error)))
}
```

```
cv=crossvalidation(y, x, split = 10, fit$lambda)
plot(log(cv$lambda), cv$MOOSE)

#### Cp ####

Cp = function(y, x, beta, lambda){

Df=fit$fit.lasso$df
l.lambda = length(lambda)
n = nrow(x)
cp=c()

for (i in 1:l.lambda){
MSE = sum((y - x %*% beta[,i])^2) / n
sigma2 = var(y - x %*% beta[,i])
cp = c(cp, MSE + 2 * Df[i] * sigma2 / n)

}

return(list(cp=cp)  )
}

Cptest=Cp(y=y, x=x, beta=fit$beta, lambda=fit$lambda)
Cptest



#### CV Benchmark ####

fit.cv = cv.glmnet(x,y)
fit.cv

fit.cv$lambda.min
```

```
#### Plot ####

plot(log(fit$lambda), fit$mse.insample, type="l", col=1)
lines(log(fit$lambda),cv$MOOSE, col=2)
lines(log(fit$lambda),Cptest$cp, col=3)
legend('topleft', legend = c('ISMSE','MOOSE','Cp'), col = 1:3, lty = 1)
```