STAT432_HW7

Taiga Hasegawa(taigah2)

2019/3/11

Question1

(a)

```
y=c(68,72)
sigma2=16
mu0=65
tau0=36
tau1=1/(1/tau0+1/sigma2)
mu1=(mu0/tau0+y[1]/sigma2)*tau1
tau2=1/(1/tau1+1/sigma2)
mu2=(mu1/tau1+y[2]/sigma2)*tau2
```

The posterior distribution of θ is N(69.0909091, 6.5454545)

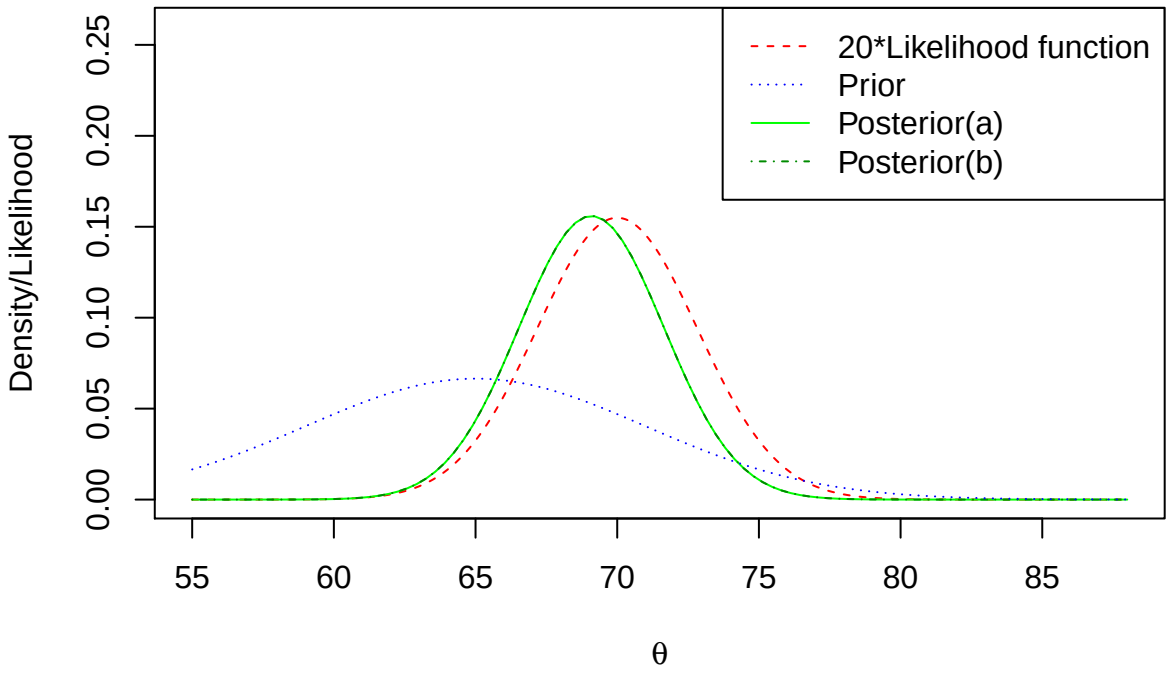
(b)

```
n=length(y)
taun=1/(1/tau0+n/sigma2)
mun=(mu0/tau0+sum(y)/sigma2)*taun
```

The posterior distribution of θ is N(69.0909091,6.5454545)

(c)

```
plot(seq(55,88,by=.1),
     sapply(seq(55,88,by=.1),function(x)20*prod(dnorm(y,x,sqrt(sigma2)))),ylim=c(0,0.26),
     type='l',lty=2,col='red',xlab=expression(theta),ylab='Density/Likelihood')
curve(dnorm(x,mu2,sd=sqrt(tau2)),from=55,to=88,col='green',add=T)
curve(dnorm(x,mun,sd=sqrt(taun)),from=55,to=88,lty=4,col='green4',add=T)
curve(dnorm(x,mu0,sd=sqrt(tau0)), lty=3, col='blue',add=T)
legend('topright',legend=c('20*Likelihood function','Prior','Posterior(a)',"Posterior(b)"),lty=c(2,3,1,
       col=c('red','blue','green',"green4"))
```



The posterior distribution I derived in (a) was identical with the one I derived in (b). These two posterior distributions were more close to likelihood fucntion.

Question2

```
data(Boston, package="MASS")
useLog = c(1,3,5,6,8,9,10,14)
Boston[,useLog] = log(Boston[,useLog])
Boston[,2] = Boston[,2] / 10
Boston[,7] = Boston[,7]^2.5 / 10^4
Boston[,11] = exp(0.4 * Boston[,11])/1000
Boston[,12] = Boston[,12] / 100
Boston[,13] = sqrt(Boston[,13])
```

```
#Ridge regression
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
X=Boston[,1:13]
y=Boston[,14]
X$chas=as.factor(X$chas)
library(glmnet)
```

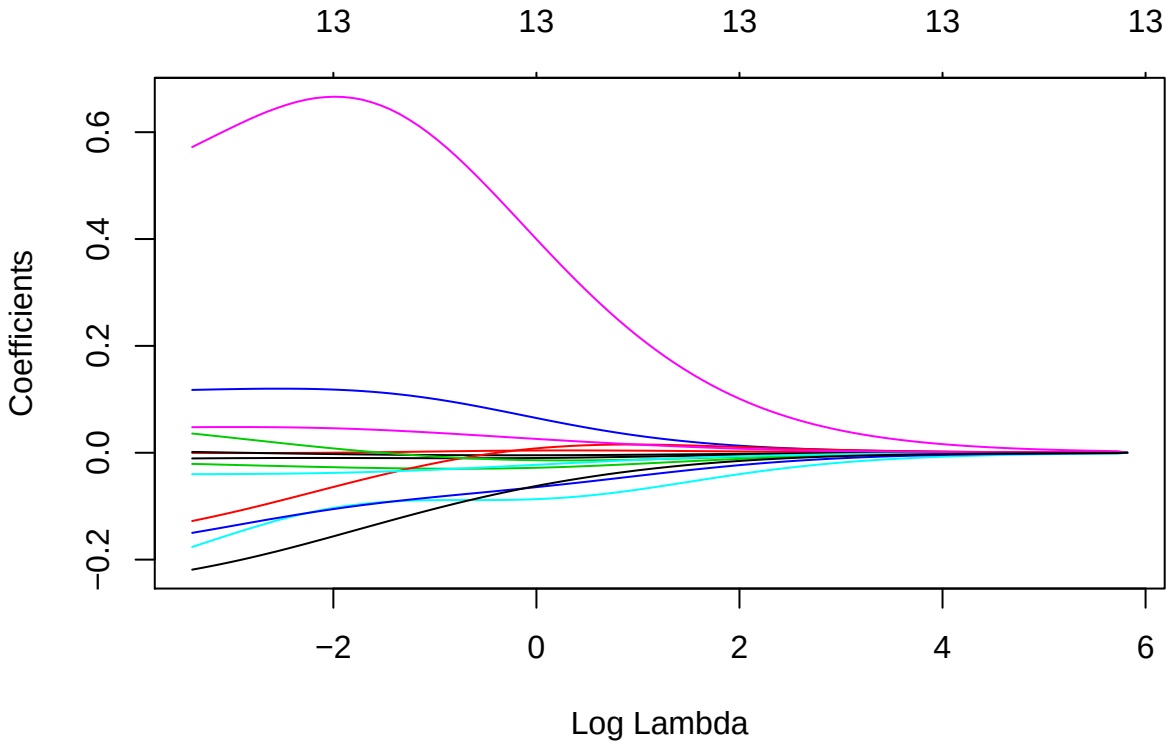
```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
set.seed(100)
#cross validation to find best lambda
lambda=0:8583
```

```
fit1 = cv.glmnet(data.matrix(X), y, nfolds = 10, alpha = 0)
plot(fit1$glmnet.fit, "lambda")
```

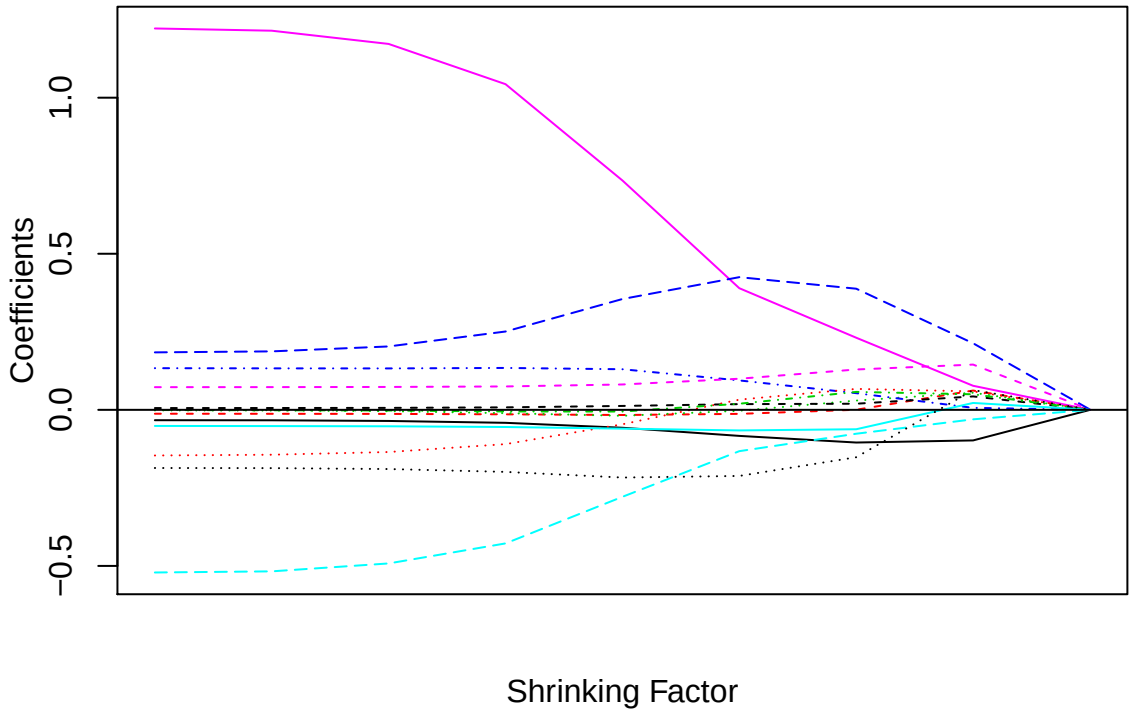



```
#Bayesian Linear Regression
library(lars)
```

```
## Loaded lars 1.2
```

```
library(mvtnorm)
library(SuppDists)
library(gtools)

X=Boston[,1:13]
n=length(y); p=dim(X)[2]
X=as.matrix(X)


Nsf = 9
lambda=c(1.5,0.9,0.4,0.2,0.1,0.05,0.03,.01,0.001)
NSamp = 11000
NBurnIn = 1000
Samp=list(2)
Samp[[1]] = array(NA,c(NSamp-NBurnIn,p,Nsf)) # beta
Samp[[2]] = array(NA,c(NSamp-NBurnIn,Nsf)) # sigma2
beta_ridge = matrix(NA,p,Nsf) # ridge estimate


# prior parameters
mu0=rep(0,p)
nu0=1; sigma20=0.5
nu_n=nu0+n
```

```r
# define rinvchisq
rinvchisq=function(n,df,scale)(df *scale)/rchisq(n, df = df)
# generate samples using Gibbs sampler: similar as coordinate descent algorithm
XTX=t(X)%*%X; XTy=t(X)%*%y
set.seed(2013)

for(sf in 1:Nsf){
  Lambda0=lambda[sf]^2*diag(p)
  beta=rmvnorm(1,mu0,Lambda0); sigma2=rinvchisq(1,nu0,sigma20)
  invLambda0=lambda[sf]^(-2)*diag(p)
  for(Iter in 1:NSamp){
    # update beta, conditioned on sigma2
    Lambda_n=solve(XTX+sigma2*invLambda0)
    mu_n=Lambda_n%*%(XTy+sigma2*invLambda0%*%mu0)
    Lambda_n=Lambda_n*sigma2
    beta=rmvnorm(1,mu_n,Lambda_n)
    # update sigma2, conditioned on beta
    sigma2_n=(nu0*sigma20+sum((y-X%*%t(beta))^2))/nu_n
    sigma2=rinvchisq(1,nu_n,sigma2_n)
    if(Iter>NBurnIn){
        Samp[[1]][Iter-NBurnIn,,sf] = beta
            Samp[[2]][Iter-NBurnIn,sf] = sigma2
    }
  }
beta_ridge[,sf] = apply(Samp[[1]][,,sf],2,median)
}


op <- par(mar=c(3,3,2,1) + 0.1,oma=rep(1,4),mgp=c(2,1,0))

matplot(t(beta_ridge),type='l',xlab='Shrinking Factor',ylab='Coefficients',xaxt='n')
abline(h=0)
```



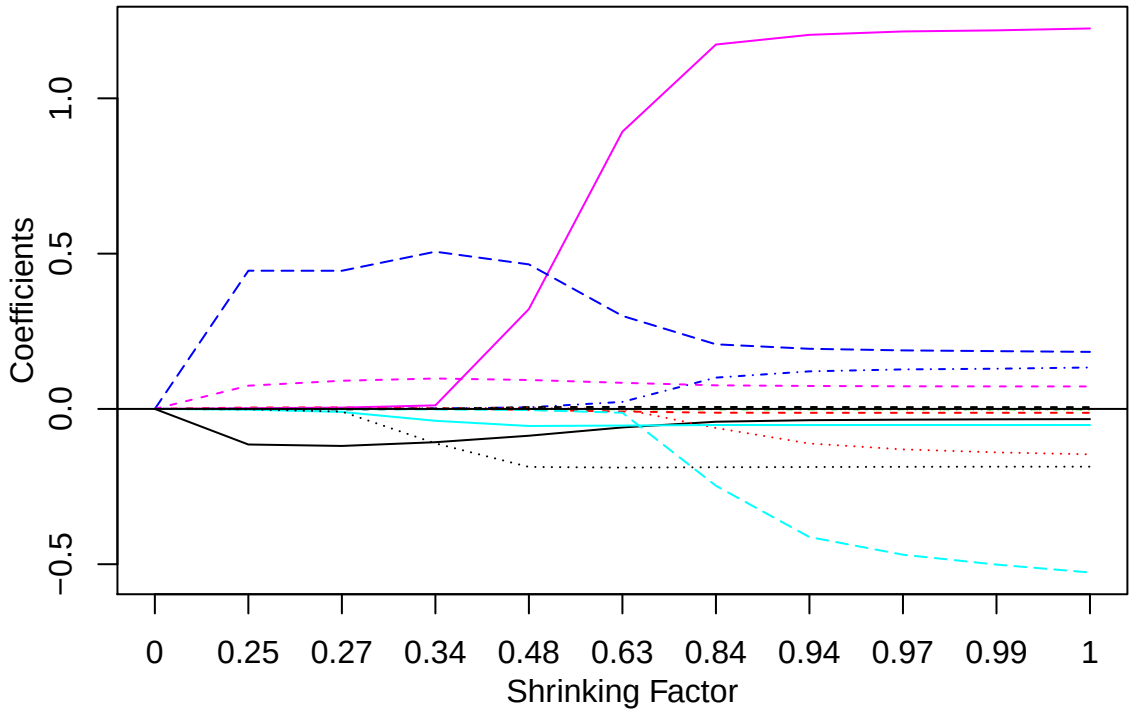

```
par(op)
```

Smaller Λ corresponds to bigger λ. Bayesian linear regression is more likely to take bigger coefficents when Λ is big enough.

Question3

```
#Bayesian Lasso
library(lars)
library(mvtnorm)
library(SuppDists) # rinvGauss
library(gtools)
N=dim(X)[1]; D=dim(X)[2]
beta_ols=lm(y~-1+X)$coefficients
OLS_NM1=norm(as.matrix(beta_ols),'1')

XX = t(X)%*%X; Xy = t(X)%*%y

# inverse Gaussian random variable generator
rInvGaussian <- function(n=1,mu,lambda){
    nv=rnorm(n)
    y=nv^2
    x=mu+mu*(mu*y-sqrt(mu*(4*lambda+mu*y)*y))/2/lambda
    z=runif(n)
    return(ifelse(z<=mu/(mu+x),x,mu^2/x))
}


## storage
Nsf = 10
```

```
lambda=c(200,140,85,42,20,4.6,1.9,.9,.4,.01)
NSamp = 11000
NBurnIn = 1000
Samp=list(3)
Samp[[1]] = array(NA,c(NSamp-NBurnIn,D,Nsf)) # beta
Samp[[2]] = matrix(NA,NSamp-NBurnIn,Nsf) # sigma2
Samp[[3]] = array(NA,c(NSamp-NBurnIn,D,Nsf)) # tau2
beta_lasso = matrix(NA,D,Nsf) # Lasso estimate


for(sf in 1:Nsf){

## Initialization
beta = rep(0.01,D)
sigma2 = 100
#tau2 = 1/rInvGaussian(1,sqrt(sigma2)*abs(lambda[sf]/beta),lambda[sf]^2)
tau2 = 1/rinvGauss(rep(1,D),sqrt(sigma2)*abs(lambda[sf]/beta),lambda[sf]^2)

for(Iter in 1:NSamp){

    # sample beta
    A = XX+diag(1/tau2); invA = solve(A)
    beta = t(rmvnorm(1,invA%*%Xy,sigma2*invA))

    # sample sigma2
      sigma2 = 1/rgamma(1,(N-1+D)/2,(sum((y-X%*%beta)^2)+sum(beta^2/tau2))/2)

    # sample tau2
  # tau2 = 1/rInvGaussian(1,sqrt(sigma2)*abs(lambda[sf]/beta),lambda[sf]^2)
      tau2 = 1/rinvGauss(rep(1,D),sqrt(sigma2)*abs(lambda[sf]/beta),lambda[sf]^2)

  # save sample beta
  if(Iter>NBurnIn){
        Samp[[1]][Iter-NBurnIn,,sf] = beta
        Samp[[2]][Iter-NBurnIn,sf] = sigma2
        Samp[[3]][Iter-NBurnIn,,sf] = tau2
  }

}
# Get Lasso estimate
#beta_lasso[,sf] = Mode(Samp[[1]][,,sf])
beta_lasso[,sf] = apply(Samp[[1]][,,sf],2,median)

}

beta_lasso=cbind(0,beta_lasso)

op <- par(mar=c(3,3,2,1) + 0.1,oma=rep(1,4),mgp=c(2,1,0))

matplot(t(beta_lasso),type='l',xlab='Shrinking Factor',ylab='Coefficients',xaxt='n')
axis(1,at=1:(Nsf+1),labels=round(apply(beta_lasso,2,function(x)norm(as.matrix(x),'1'))/OLS_NM1,2))
abline(h=0)
```




```
par(op)
```

```
#Frequentist
set.seed(100)
fit2 = cv.glmnet(data.matrix(X), y, nfolds = 10, alpha = 1,family = "gaussian")
plot(fit2$glmnet.fit, "lambda")
```