HW 4 R Code

Problem 1 R Code

#Problem 1 Part a

set.seed(1)

t=10000

azdiabetes = read.table("azdiabetes.dat", header = TRUE);

y=as.matrix(azdiabetes[,2])

g=length(y)

X=as.matrix(cbind(rep(1,g),azdiabetes[,-c(2,8)]))

n=nrow(X)

m=ncol(X)

nu\_0=2

sigma2\_0=1

Hg=(g/(g+1))\*X%\*%solve(t(X)%\*%X)%\*%t(X)

SS=t(y)%\*%(diag(1,nrow=n)-Hg)%\*%y

sigma2=1/rgamma(t,(nu\_0+n)/2,(nu\_0\*sigma2\_0+SS)/2)

Vb=g\*solve(t(X)%\*%X)/(g+1)

Eb=Vb%\*%t(X)%\*%y

E=matrix(rnorm(t\*m,0,sqrt(sigma2)),t,m)

b=t(t(E%\*%chol(Vb))+c(Eb))

for (i in 1:m)

{

print(quantile(b[,i],c(0.025, 0.975)))

}

plot(density(b[,2]),col=1, xlab=names(azdiabetes)[1],

main ="Problem 1 Part a")

abline(v=mean(b[,2]),col=2)

abline(v=quantile(b[,2],0.025),col=4)

abline(v=quantile(b[,2],0.975),col=4)

for (i in 2:6)

{

plot(density(b[,i+1]),col=1, xlab=names(azdiabetes)[i+1],

main ="Problem 1 Part a")

abline(v=mean(b[,i+1]),col=2)

abline(v=quantile(b[,i+1],0.025),col=4)

abline(v=quantile(b[,i+1],0.975),col=4)

}

#Problem 1 Part b

lpy.X=function(y, X)

{

n=nrow(X)

m=ncol(X)

g=length(y)

nu\_0=1

sigma2\_0=try(summary(lm(y ~ -1+X))$sigma^2, silent = TRUE)

if (m == 0)

{

Hg=0

sigma2\_0=mean(y^2)

}

else if (m > 0)

{

Hg=(g/(g+1))\*X%\*%solve(t(X)%\*%X)%\*%t(X)

}

SS=t(y)%\*%(diag(1, nrow = n)-Hg)%\*%y

-(1/2)\*(n\*log(pi)+m\*log(1+g)+(nu\_0+n)\*log(nu\_0\*sigma2\_0+SS)-

nu\_0\*log(nu\_0\*sigma2\_0))+lgamma((nu\_0+n)/2)-lgamma(nu\_0/2)

}

t=1000

A=matrix(NA, t, m)

B=matrix(0, t, m)

a=rep(1, m)

lpy.c=lpy.X(y, X[, a == 1, drop = FALSE])

for(i in 1:t)

{

for(j in sample(1:m))

{

temp=a

temp[j]=1-temp[j]

lpy.m=lpy.X(y, X[, temp == 1, drop = FALSE])

r=(lpy.m-lpy.c)\*(-1)^(temp[j] == 0)

a[j]=rbinom(1, 1, 1/(1+exp(-r)))

if (a[j] == temp[j])

{

lpy.c=lpy.m

}

A[i, ]=a

Hg=(g/(g+1))\*X[, A[i,] == 1, drop = FALSE]%\*%solve(t(X[,

A[i,] == 1, drop = FALSE])%\*%X[, A[i,] == 1,

drop = FALSE])%\*%t(X[, A[i,] == 1, drop = FALSE])

SS=t(y)%\*%(diag(1, nrow = n)-Hg)%\*%y

sigma2=1/rgamma(t, (nu\_0+n)/2, (nu\_0\*sigma2\_0+SS)/2)

Vb=g\*solve(t(X[, A[i,] == 1, drop = FALSE])%\*%X[, A[i,] == 1,

drop = FALSE])/(g+1)

Eb=Vb%\*%t(X[, A[i,] == 1, drop = FALSE])%\*%y

E=matrix(rnorm(sum(A[i, ]), 0, sqrt(sigma2)), 1, sum(A[i, ]))

B[i, A[i,] == 1]=t(t(E%\*%chol(Vb))+c(Eb))

}

}

for (i in 1:m)

{

print(sum(B[,i]!= 0)/t)

}

for (i in 1:m)

{

print(quantile(B[,i],c(0.025, 0.975)))

}

plot(density(B[,2]),col=1, xlab=names(azdiabetes)[1],

main ="Problem 1 Part b")

abline(v=mean(B[,2]),col=2)

abline(v=quantile(B[,2],0.025),col=4)

abline(v=quantile(B[,2],0.975),col=4)

for (i in 2:6)

{

plot(density(B[,i+1]),col=1, xlab=names(azdiabetes)[i+1],

main ="Problem 1 Part b")

abline(v=mean(B[,i+1]),col=2)

abline(v=quantile(B[,i+1],0.025),col=4)

abline(v=quantile(B[,i+1],0.975),col=4)

}

Problem 2 Code

#Problem 2 Part a

crime = read.table("crime.dat", header = TRUE)

y = as.matrix(crime[,1])

X = as.matrix(cbind(rep(1, dim(y)[1], 1),crime[,-1]))

g = length(y)

nu\_0 = 2

sigma2\_0 = 1

n = dim(X)[1]

m = dim(X)[2]

t = 10000

Hg = (g/(g+1))\*X%\*%solve(t(X)%\*%X)%\*%t(X)

SSR = t(y)%\*%(diag(1, nrow = n)-Hg)%\*%y

sigma2 = 1/rgamma(t, (nu\_0+n)/2, (nu\_0\*sigma2\_0+SSR)/2)

Vb = g\*solve(t(X)%\*%X)/(g+1)

Eb = Vb%\*%t(X)%\*%y

E = matrix(rnorm(t\*m, 0, sqrt(sigma2)), t, m)

beta = t(t(E%\*%chol(Vb))+c(Eb));

par(mfrow = c(2, 2))

beta\_mean=numeric(m)

beta\_l=numeric(m)

beta\_u=numeric(m)

for (i in 1:m) {

plot(density(beta[,i]), col = 2, lwd = 1,

xlab = names(crime)[i], main = "Problem 2 part a");

abline(v = quantile(beta[,i], c(0.025, 0.975))[1],

col = 3, lwd = 2);

abline(v = quantile(beta[,i], c(0.025, 0.975))[2],

col = 3, lwd = 2);

abline(v = mean(beta[,i]), col = 4, lwd = 2);

beta\_mean[i]=mean(beta[,i])

beta\_l[i]=quantile(beta[,i], c(0.025, 0.975))[1]

beta\_u[i]=quantile(beta[,i], c(0.025, 0.975))[2]

}

beta\_mean

beta\_l

beta\_u

# Least squares estimates

LS = lm(y ~ M+So+Ed+Po1+Po2+LF+M.F+Pop+NW+U1+U2+GDP+Ineq+Prob+Time,

data = crime)

summary(LS)

MSE = sum(LS$residuals^2)/(n-16);

#upper 95% bound

LS$coefficients+qt(1-0.05/2,df=n-16)\*sqrt(diag(MSE\*solve(t(X)%\*%X)))

#lower 95% bound

LS$coefficients-qt(1-0.05/2,df=n-16)\*sqrt(diag(MSE\*solve(t(X)%\*%X)))

#Problem 2 Part b (i)

>

> set.seed(1)

> tr = sample(1:n, n/2)

> te = (1:n)[-tr]

> X\_tr = X[tr,]

> X\_te = X[te,]

> y\_tr = y[tr,]

> y\_te = y[te,]

> beta\_ols = solve(t(X\_tr)%\*%X\_tr)%\*%t(X\_tr)%\*%y\_tr

> y\_ols = X\_te%\*%beta\_ols

> dev.off()

> plot(y\_te, y\_ols, xlab = "y\_te", ylab = "y\_ols",

main='Problem 2 part b (i)', xlim = c(-2, 4), ylim = c(-2, 4),

asp = 1);

> abline(c(0, 0), c(1, 1), col = "red")

> sum((y\_ols-y\_te)^2)/length(y\_te)

[1] 0.5562224

> #Problem 2 Part b (ii)

>

> g = length(y\_tr);

> nu\_0 = 2;

> sigma2\_0 = 1;

> n = dim(X\_tr)[1];

> m = dim(X\_tr)[2];

> t = 10000;

> Hg = (g/(g+1))\*X\_tr%\*%solve(t(X\_tr)%\*%X\_tr)%\*%t(X\_tr);

> SSR = t(y\_tr)%\*%(diag(1, nrow = n)-Hg)%\*%y\_tr;

> sigma2 = 1/rgamma(t, (nu\_0+n)/2, (nu\_0\*sigma2\_0+SSR)/2);

> Vb = g\*solve(t(X\_tr)%\*%X\_tr)/(g+1);

> Eb = Vb%\*%t(X\_tr)%\*%y\_tr;

> E = matrix(rnorm(t\*m, 0, sqrt(sigma2)), t, m);

> beta\_bayes = colMeans(t(t(E%\*%chol(Vb))+c(Eb)));

> y\_bayes = X\_te%\*%beta\_bayes;

> plot(y\_te, y\_bayes, xlab = "y\_te", ylab = "y\_bayes",

main='Problem 2 part b (ii)', xlim = c(-2, 4), ylim = c(-2, 4),

asp = 1);

> abline(c(0, 0), c(1, 1), col = "red");

> sum((y\_bayes-y\_te)^2)/length(y\_te);

[1] 0.5218922

> #Problem 2 part c

> t1 = 10000;

> error\_ols = rep(NA, 1, t1);

> error\_bayes = rep(NA, 1, t1);

> for (i in 1:t1) {

+ set.seed(i);

+ n = 47;

+ tr = sample(1:n, n/2);

+ te = (1:n)[-tr];

+ X\_tr = X[tr,];

+ X\_te = X[te,];

+ y\_tr = y[tr,];

+ y\_te = y[te,];

+ beta\_ols = solve(t(X\_tr)%\*%X\_tr)%\*%t(X\_tr)%\*%y\_tr;

+ y\_ols = X\_te%\*%beta\_ols;

+ error\_ols[i] = sum((y\_ols-y\_te)^2)/length(y\_te)

+ g = length(y\_tr);

+ nu\_0 = 2;

+ sigma2\_0 = 1;

+ n = dim(X\_tr)[1];

+ m = dim(X\_tr)[2];

+ t = 10000;

+ Hg = (g/(g+1))\*X\_tr%\*%solve(t(X\_tr)%\*%X\_tr)%\*%t(X\_tr);

+ SSR = t(y\_tr)%\*%(diag(1, nrow = n)-Hg)%\*%y\_tr;

+ sigma2 = 1/rgamma(t, (nu\_0+n)/2, (nu\_0\*sigma2\_0+SSR)/2);

+ Vb = g\*solve(t(X\_tr)%\*%X\_tr)/(g+1);

+ Eb = Vb%\*%t(X\_tr)%\*%y\_tr;

+ E = matrix(rnorm(t\*m, 0, sqrt(sigma2)), t, m);

+ beta\_bayes = colMeans(t(t(E%\*%chol(Vb))+c(Eb)));

+ y\_bayes = X\_te%\*%beta\_bayes;

+ error\_bayes[i] = sum((y\_bayes-y\_te)^2)/length(y\_te);

+ }

> mean(error\_ols)

[1] 1.216266

> mean(error\_bayes)

[1] 1.145592

Problem 3 Code

#Problem 3 Part b

msparrownest <- read.table("msparrownest.dat", quote="\"",

comment.char="")

y=msparrownest$V1

x=msparrownest$V2

f1=glm(formula = y~x,family = binomial, data=msparrownest)

mu.alpha=summary(f1)$coefficients[1,1]

mu.alpha

sd.alpha=summary(f1)$coefficients[1,2]

sd.alpha

mu.beta=summary(f1)$coefficients[2,1]

mu.beta

sd.beta=summary(f1)$coefficients[2,2]

sd.beta

#Problem 3 Part c

posterior=function(alpha,beta,y,x){

b=sum(log(1+exp(alpha+beta\*x)))

theta=alpha+beta\*x

likelihood=exp(sum(y\*theta)-b)

prior.alpha=dnorm (alpha, mean=mu.alpha, sd=sd.alpha)

prior.beta=dnorm (beta, mean=mu.beta, sd=sd.beta)

return(likelihood\*prior.alpha\*prior.beta)

}

#Metropolis

B=20000

n=10000000

alpha.m=rep(NA,B+n)

beta.m=rep(NA,B+n)

alpha.m[1]=mu.alpha

beta.m[1]=mu.beta

acc=0

for ( t in 2:(B+n)){

alpha.new=rnorm(1,mean=alpha.m[t-1],sd=.1)

beta.new=rnorm(1,mean=beta.m[t-1],sd=.1)

pi.new=posterior(alpha.new, beta.new, y,x)

pi.t=posterior(alpha.m[t-1], beta.m[t-1], y,x)

rho=min(c(pi.new/pi.t,1))

U=runif (1 ,min=0,max=1)

if (U<rho){

alpha.m[t]=alpha.new

beta.m[t]=beta.new

acc=acc+1

} else{

alpha.m[t]=alpha.m[t-1]

beta.m[t]=beta.m[t-1]

}

}

print(acc/(B+n))

effectiveSize(alpha.m)

effectiveSize(beta.m[1:((B+n)/2)]) +

effectiveSize(beta.m[((B+n)/2+1):(B+n)])

#Problem 3 Part d

grapha=density(alpha.m)

graphb=density(beta.m)

prior.alpha=dnorm(grapha$x, mean=mu.alpha, sd=sd.alpha)

prior.beta=dnorm(graphb$x, mean=mu.beta, sd=sd.beta)

plot(grapha$x,grapha$y,lwd=2,typ="l", xlab = "alpha",

ylab = "density", main = "Prior and Posterior for alpha")

lines(grapha$x,prior.alpha,col="blue",lwd=2)

legend("topright",legend = c("prior","posterior"),lwd=c(1,1),

col=c("blue","black"))

plot(graphb$x,graphb$y,typ="l",xlab = "beta",ylab = "density",

main = "Prior and Posterior for beta")

lines(graphb$x,prior.beta,col="blue")

legend("topright",legend = c("prior","posterior"),lwd=c(1,1),

col=c("blue","black"))

#Problem 3 Part e

x.seq=seq(10,15,by=0.02)

conf.band=matrix(NA,nrow=length(x.seq),ncol=2)

for (i in 1:length(x.seq)) {

fun=exp(alpha.m+beta.m\*x.seq[i])

fun1=fun/(1+fun)

conf.band[i,]=quantile(fun1,probs=c(0.025, 0.975), na.rm = TRUE)

}

plot(x.seq,conf.band[,1],ylim=c(0,1),lwd=2,col="black",type="l",

main="Confidence Band",xlab="x",ylab="Percentile")

points(x.seq,conf.band[,2],col="red",type="l")

legend("topleft", col=c("black","red"),

legend=c("lower quantile","upper quantile"), lwd = c(1,1))

Problem 4 Code

#Problem 4 Part 1

library(LearnBayes)

library(MASS)

data(donner)

y=as.matrix(donner$survival,ncol=1)

X=cbind(1,donner$age,donner$male)

n=dim(X)[1]

p=dim(X)[2]

X=as.matrix(X,nrow=n,ncol=p)

Sigma=solve(diag(x=1,nrow=p,ncol=p)/100+t(X) %\*% X)

#Gibbs

B=1000

nmc=10000

beta.mc=matrix(NA,nrow=p,ncol=nmc+B)

z.mc=matrix(NA,nrow=n,ncol=nmc+B)

fit.freq=glm(donner$survival~X-1,family=binomial)

beta.mc[,1]=fit.freq$coefficients

z.mc[,1]=X%\*%beta.mc[,1]

zi.cond<-function(zi,xi,beta,sigma,yi){

a=dnorm(zi,mean=t(xi)%\*%beta,sd=sigma)\*exp(yi\*zi-log(1+exp(zi)))

return(a)

}

for (t in 2:(B+nmc)){

mu.beta=Sigma %\*% t(X) %\*% as.matrix(z.mc[,t-1],ncol=1)

beta.mc[,t]=mvrnorm(1,mu=mu.beta,Sigma)

pz.s=mvrnorm(1,mu=z.mc[,t-1],Sigma=diag(x=1,nrow=n,ncol=n))

for (i in 1:n){

xi=as.matrix(X[i,],ncol=1)

beta=as.matrix(beta.mc[,t],ncol=1)

yi=y[i]

pz.s=zi.cond(pz.s[i],xi,beta,sigma=1,yi)

pz.t=zi.cond(z.mc[i,t-1],xi,beta,sigma=1,yi)

rho=min(c(1,pz.s/pz.t))

if (runif(1,min=0,max=1)<rho)

{ z.mc[i,t]=pz.s[i]

}else{

z.mc[i,t]=z.mc[i,t-1]

}

}

}

#Posterior results for logistic model

beta.mc=beta.mc[,(B+1):(B+nmc)]

beta.mc1.mean=apply(beta.mc,1,mean)

beta.mc1.lower=apply(beta.mc,1,quantile,.025)

beta.mc1.upper=apply(beta.mc,1,quantile,.975)

print(beta.mc1.mean)

print(beta.mc1.upper)

print(beta.mc1.lower)

#Probit model Results

beta.probit=bayes.probit(y,X,m=(nmc+B),prior=list(beta=c(0,0,0),

P=diag(x=1,nrow=p,ncol=p)/100))$beta

beta.probit=beta.probit[(B+1):(B+nmc),]

beta.probit.mean=apply(beta.probit,2,mean)

beta.probit.lower=apply(beta.mc,1,quantile,.025)

beta.probit.upper=apply(beta.mc,1,quantile,.975)

print(beta.probit.mean)

print(beta.probit.upper)

print(beta.probit.lower)

#Problem 4 Part 2

B=2000

nmc=20000

beta.mc=matrix(NA,nrow=p,ncol=nmc+B)

z.mc=matrix(NA,nrow=n,ncol=nmc+B)

beta.mc[,1]=fit.freq$coefficients

z.mc[,1]=X%\*%beta.mc[,1]

sigma2.mc=rep(NA,B+nmc)

sigma2.mc[1]=1

count=rep(0,n)

#Gibbs Sampling:

for (t in 2:(B+nmc)){

Sigma.beta=solve(diag(x=1,nrow=p,ncol=p)/100+t(X) %\*%

X/sigma2.mc[t-1])

mu.beta=Sigma.beta %\*% t(X) %\*%as.matrix(z.mc[,t-1],

ncol=1)/sigma2.mc[t-1]

beta.mc[,t]=mvrnorm(1,mu=mu.beta,Sigma.beta)

pz.s=mvrnorm(1,mu=z.mc[,t-1],Sigma=0.01\*diag(1,ncol=n,nrow=n))

for (i in 1:n){

xi=as.matrix(X[i,],ncol=1)

beta=as.matrix(beta.mc[,t],ncol=1)

yi=y[i]

pz.s=zi.cond(pz.s[i],xi,beta,sigma=sqrt(sigma2.mc[t-1]),yi)

pz.t=zi.cond(z.mc[i,t-1],xi,beta,sigma=sqrt(sigma2.mc[t-1]),yi)

rho=min(c(1,pz.s/pz.t))

if (runif(1,min=0,max=1)<rho){

z.mc[i,t]=pz.s[i]

count[i]=count[i]+1

}

else{z.mc[i,t]=z.mc[i,t-1]}

}

SSR.t = sum((z.mc[,t] - X %\*% as.matrix(beta.mc[,t],ncol=1))^2)

sigma2.mc[t] = 1/rgamma(1,shape=0.1+n/2,rate=1+SSR.t/2)

}

beta.mc=beta.mc[,(B+1):(B+nmc)]

#Posterior results for logistic model

beta.mc2.mean=apply(beta.mc,1,mean)

beta.mc2.upper=apply(beta.mc,1,quantile,.975)

beta.mc2.lower=apply(beta.mc,1,quantile,.025)

print(beta.mc2.mean)

print(beta.mc2.upper)

print(beta.mc2.lower)

#Probit model Results

beta.probit=bayes.probit(y,X,m=(nmc+B),prior=list(beta=c(0,0,0),

P=diag(x=1,nrow=p,ncol=p)/100))$beta

beta.probit=beta.probit[(B+1):(B+nmc),]

beta.probit.mean=apply(beta.probit,2,mean)

beta.probit.upper=apply(beta.mc,1,quantile,.975)

beta.probit.lower=apply(beta.mc,1,quantile,.025)

print(beta.probit.mean)

print(beta.probit.upper)

print(beta.probit.lower)