Exam Solutions

Ahmed Alhasan 'ahmal787'

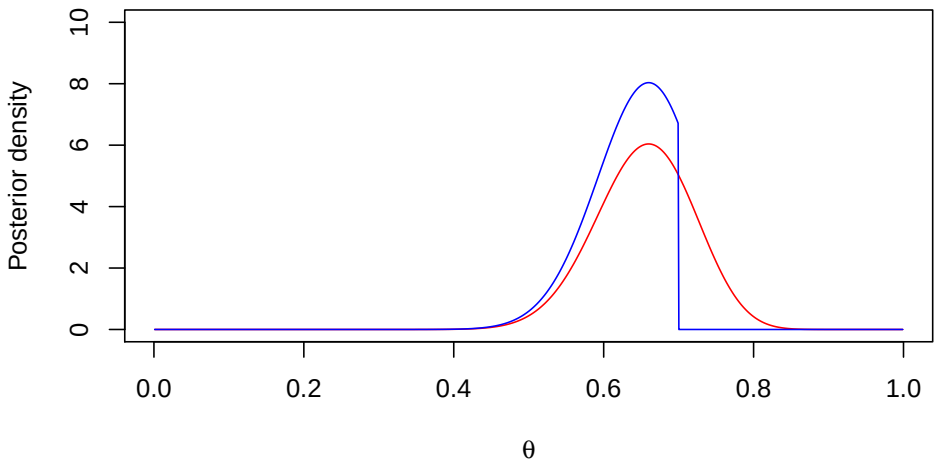
2020-06-04

Q1

- (a)
- (b)

```
x = 33


logPostTheta <- function(x, theta, a, b){
  sum(dunif(theta,min=a,max = b, log = TRUE) + dbinom(x,size = 50, prob = theta, log = TRUE))
}

a = 0
b = 1

thetaGrid <- seq(0.001, 0.999, length = 1000)
logPostEvals_1 <- rep(0, 1000)
i = 0
for (theta in thetaGrid){
  i = i + 1
  logPostEvals_1[i] <- logPostTheta(x, theta, a, b)
}

a = 0.3
b = 0.7
logPostEvals_2 <- rep(0, 1000)
i = 0
for (theta in thetaGrid){
  i = i + 1
  logPostEvals_2[i] <- logPostTheta(x, theta, a, b)
}


binWidth = thetaGrid[2]-thetaGrid[1]
logPostEvals <- data.frame(Prior_1 = exp(logPostEvals_1)/(sum(exp(logPostEvals_1))*binWidth),
                           Prior_2 = exp(logPostEvals_2)/(sum(exp(logPostEvals_2))*binWidth))


plot(thetaGrid, logPostEvals$Prior_1, type = "l",
     ylab = 'Posterior density',
     xlab = expression(theta),
```

```
     col = "red", ylim = c(0,10))
lines(thetaGrid, logPostEvals$Prior_2, type = "l",
      ylab = 'Posterior density',
      xlab = expression(theta),
      col = "blue")
```



- (c)

```
cat("Prior_1: " , mean(logPostEvals$Prior_1<.5))
```

```
## Prior_1:  0.708
```

```
cat("Prior_2: " , mean(logPostEvals$Prior_2<.5))
```

```
## Prior_2:  0.795
```

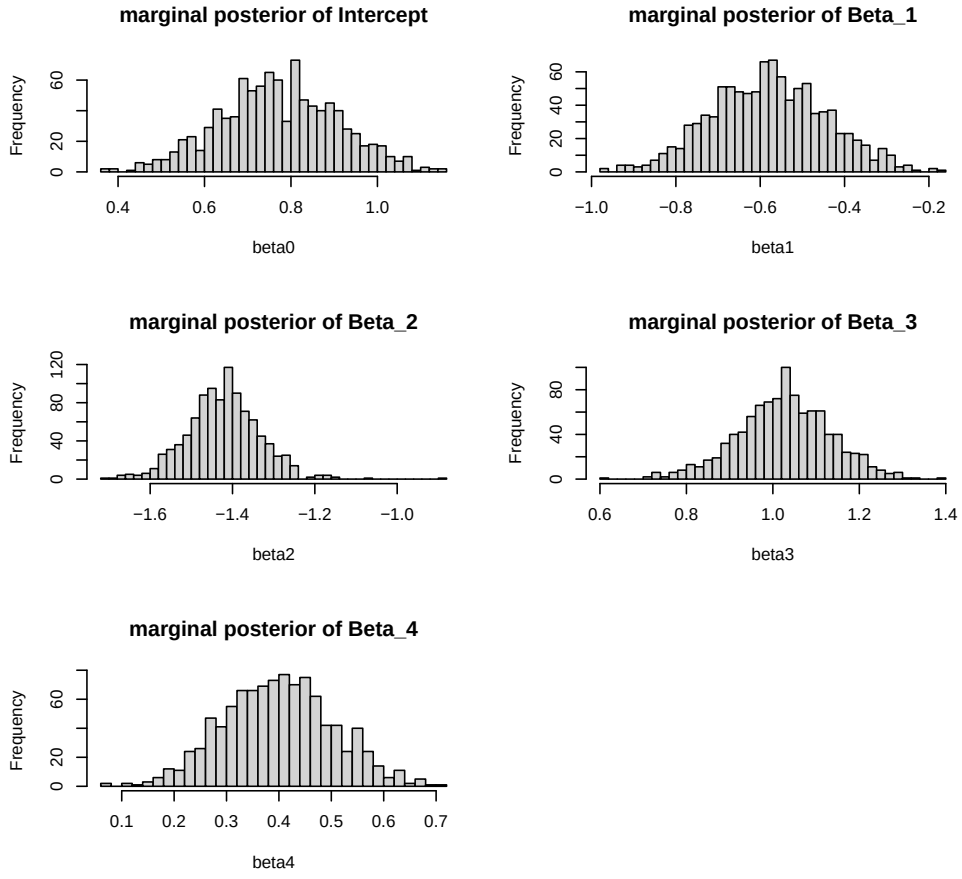
Q2

- (a)

```
setwd("C:/Machine_Learning/1_Workshop/9_Bayesian_Learning/3_Exams/10_2020_Jun_04")
load(file = 'titanic.RData')

if("mvtnorm" %in% rownames(installed.packages()) == FALSE) {install.packages("mvtnorm")}
if("msm" %in% rownames(installed.packages()) == FALSE) {install.packages("msm")}
library(mvtnorm) # For mulitvariate normal
library(msm) # For truncated normal

BayesProbReg <- function(y, X, mu_0, tau, nIter){
  # Gibbs sampling in probit regression using data augmentation:
```

```
  #
  # beta | tau ~ N(mu_0, tau^2*I)
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones if you want an intercept.
  #   mu_0 - prior mean for beta
  #   tau - prior standard deviation for beta
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   betaSample    - Posterior samples of beta.          nIter-by-nCovs matrix

  # Prior
  nPara <- dim(X)[2]
  priorCov <- tau^2*diag(nPara)
  priorPrec <- solve(priorCov)

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  n1 = sum(y)
  n0 = n - n1
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X

  # The actual sampling
  betaSample = matrix(NA, nIter, nCovs)
  u <- matrix(NA, n, 1)
  beta <- solve(XX,crossprod(X,y)) # OLS estimate as initial value
  for (i in 1:nIter){

    xBeta <- X%*%beta

    # Draw u | beta
    u[y == 0] <- rtnorm(n = n0, mean = xBeta[y==0], sd = 1, lower = -Inf, upper = 0)
    u[y == 1] <- rtnorm(n = n1, mean = xBeta[y==1], sd = 1, lower = 0, upper = Inf)

    # Draw beta | u
    betaHat <- solve(XX,t(X)%*%u)
    postPrec <- XX + priorPrec
    postCov <- solve(postPrec)
    betaMean <- solve(postPrec,XX%*%betaHat + priorPrec%*%mu_0)
    beta <- t(rmvnorm(n = 1, mean = betaMean, sigma = postCov))
    betaSample[i,] <- t(beta)

  }

  return(betaSample=betaSample)
}



y <- titanic[,1]
```

```r
X <- as.matrix(titanic[,-1])
mu_0 <- rep(0, dim(X)[2])
tau  <- 50
nIter <- 1000

betaSample <- BayesProbReg(y, X, mu_0, tau, nIter)

par(mfrow = c(3,2))
hist(betaSample[,1], breaks = 40, main = "marginal posterior of Intercept", xlab = expression(beta0))
hist(betaSample[,2], breaks = 40, main = "marginal posterior of Beta_1", xlab = expression(beta1))
hist(betaSample[,3], breaks = 40, main = "marginal posterior of Beta_2", xlab = expression(beta2))
hist(betaSample[,4], breaks = 40, main = "marginal posterior of Beta_3", xlab = expression(beta3))
hist(betaSample[,5], breaks = 40, main = "marginal posterior of Beta_4", xlab = expression(beta4))
```



- (b)
- (c)

```r
cat("Probablity of Beta2 + Beta5>0: ", mean(betaSample[,2] + betaSample[,5] > 0))
```

```
## Probablity of Beta2 + Beta5>0:  0.14
```

- (d)

Q3

- (a)

```
x <- c(log(20), log(20), log(50), log(40))
y <- c(5, 3, 17, 8)
logPostBeta <- function(beta, x, y){
  PostBeta  <- vector(length = 4)
  for(i in 1:length(x)){
    bPrior <- dnorm(x[i], 1, 0.1, log = TRUE)
    logLike <- sum(dpois(y[i], lambda = exp(x*beta), log = TRUE))
    PostBeta[i] <-  bPrior + logLike
  }
  return(PostBeta)
}

initVal <- as.vector(rep(0,length(x)))
OptimResults <-optim(initVal,logPostBeta, gr=NULL, x, y,method=c("BFGS"),
                     control=list(fnscale=-1),hessian=TRUE)
```

- (b)

Q4

- (c)

```
prob_fm <- dnorm(10, 14, 2) * dnorm(250, 300, 50) * 5/22
prob_fm
```

```
## [1] 2.969144e-05
```