Solution to computer exam in Bayesian learning

Per Siden

2020-10-22

First load all the data into memory by running the R-file given at the exam

```
rm(list=ls())
source("ExamData.R")
set.seed(1)
```

Problem 1

This is a Bernoulli likelihood with a $Beta(1,1)$ prior, which means the posterior is $Beta(s+1, f+1)$, where s is the number of correctly classified images and $f = n - s$, in each case.

1a

```
n = 100
s1 = 95
f1 = n-s1
s2 = 87
f2 = n-s2

c(1 - pbeta(0.9,s1+1,f1+1),1 - pbeta(0.9,s2+1,f2+1))
```

```
## [1] 0.9458097 0.1313070
```

The posterior probabilites are 0.95 and 0.13.

1b

```
N <- 10000
sim1 <- rbeta(N,s1+1,f1+1)
sim2 <- rbeta(N,s2+1,f2+1)

mean(sim1>sim2)
```

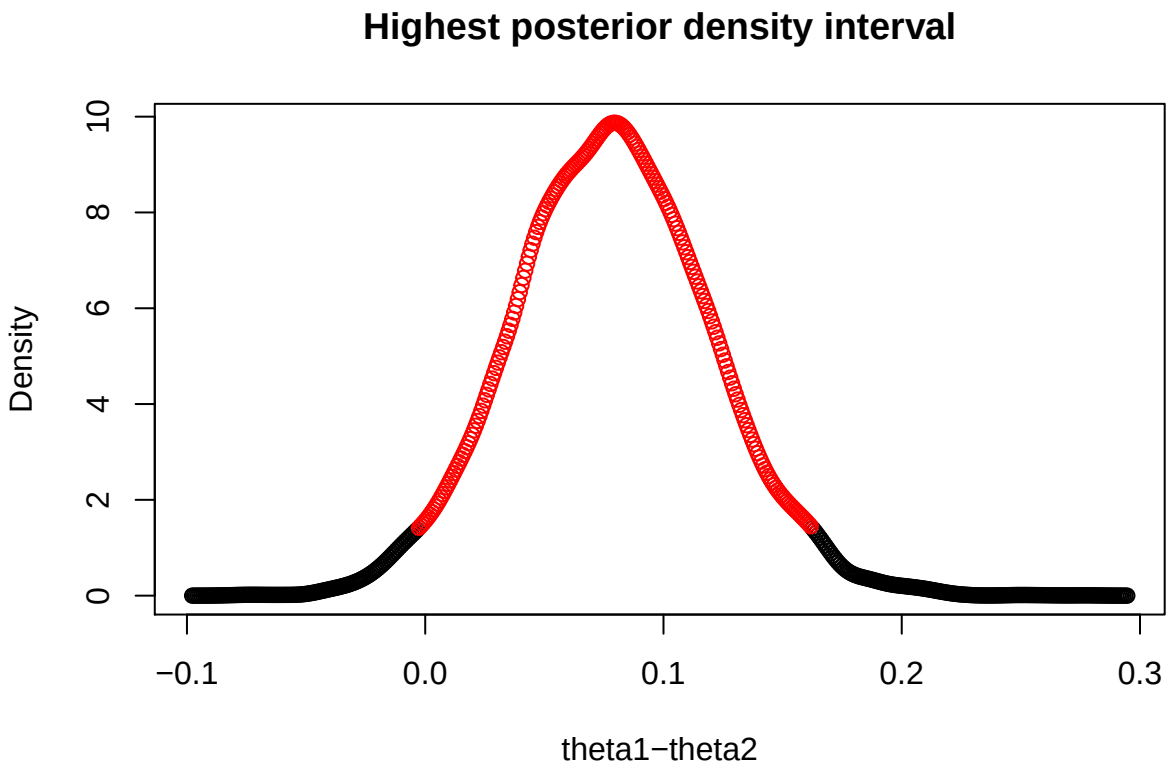
```
## [1] 0.9722
```

The posterior probability that $\theta_1 > \theta_2$ is 0.97, which means that there is a 97% probability that C_1 is a better classifier than C_2, in terms of accuracy.

1c

```
simDiff = sim1 - sim2
dD = density(simDiff)
dn=sort(dD$y/sum(dD$y),index.return=TRUE);
dnn = cumsum(dn$x)
HPDind = sort(dn$ix[dnn > .05])
notHPDind = sort(dn$ix[dnn <= .05])
plot(dD$x[notHPDind],dD$y[notHPDind],xlim=c(min(dD$x),max(dD$x)),
     ylim=c(min(dD$y),max(dD$y)),title("Highest posterior density interval"),
```

```
     xlab="theta1-theta2",ylab="Density")
points(dD$x[HPDind],dD$y[HPDind],col=2)
```




```
CI = c(min(dD$x[HPDind]),max(dD$x[HPDind]))
cat("\nHighest posterior density interval: ",CI)
```

```
##
## Highest posterior density interval:  -0.002575527 0.1617891
```

The HPD interval is (0.00,0.16).

1d

This probability can be computed as $E(x^*) = E(E(x^*|\theta)|x_1, \ldots, x_{100}) = E(\theta|x_1, \ldots, x_{100})$, that is the posterior mean of θ.

```
c((s1+1)/(n+1+1),(s2+1)/(n+1+1))
```

```
## [1] 0.9411765 0.8627451
```

The posterior predictive probabilities are 0.94 and 0.86.

Problem 2

2a and 2b on paper.

2c

```
n <- 3
x1 <- 9
x2 <- 1
x3 <- 9
```

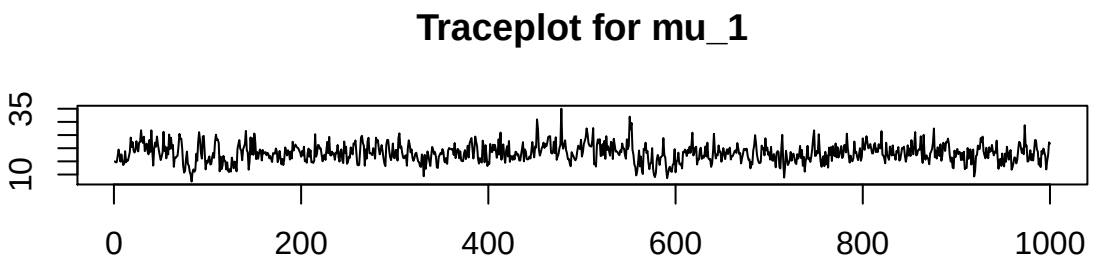
```
alpha <- 1
beta <- 1
S = x1+x2+x3

mlike <- function(K){
  num <- choose(K,x1)*choose(K,x2)*choose(K,x3)*gamma(alpha+beta)*gamma(S+alpha)*gamma(n*K-S+beta)
  den <- gamma(alpha)*gamma(beta)*gamma(alpha+beta+n*K)
  return(num/den)
}

mlike1 <- mlike(10)
mlike2 <- mlike(20)

postprob <- c(mlike1,mlike2)/(mlike1+mlike2)
print(postprob)
```

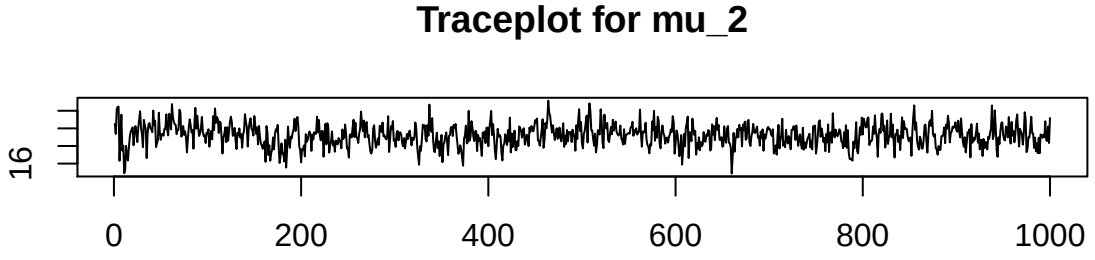
```
## [1] 0.1154727 0.8845273
```

Problem 3

3a

```
x <- listeners
nComp <- 2
nIter <- 1000
set.seed(100)

gibbs <- GibbsMixNormal(x, nComp, nIter)

par(mfrow=c(2,1))
plot(sqrt(gibbs$sigma2Sample[,1]), type="l", main="Traceplot for mu_1",xlab="",ylab="")
plot(gibbs$muSample[,2], type="l", main="Traceplot for mu_2",xlab="",ylab="")
```



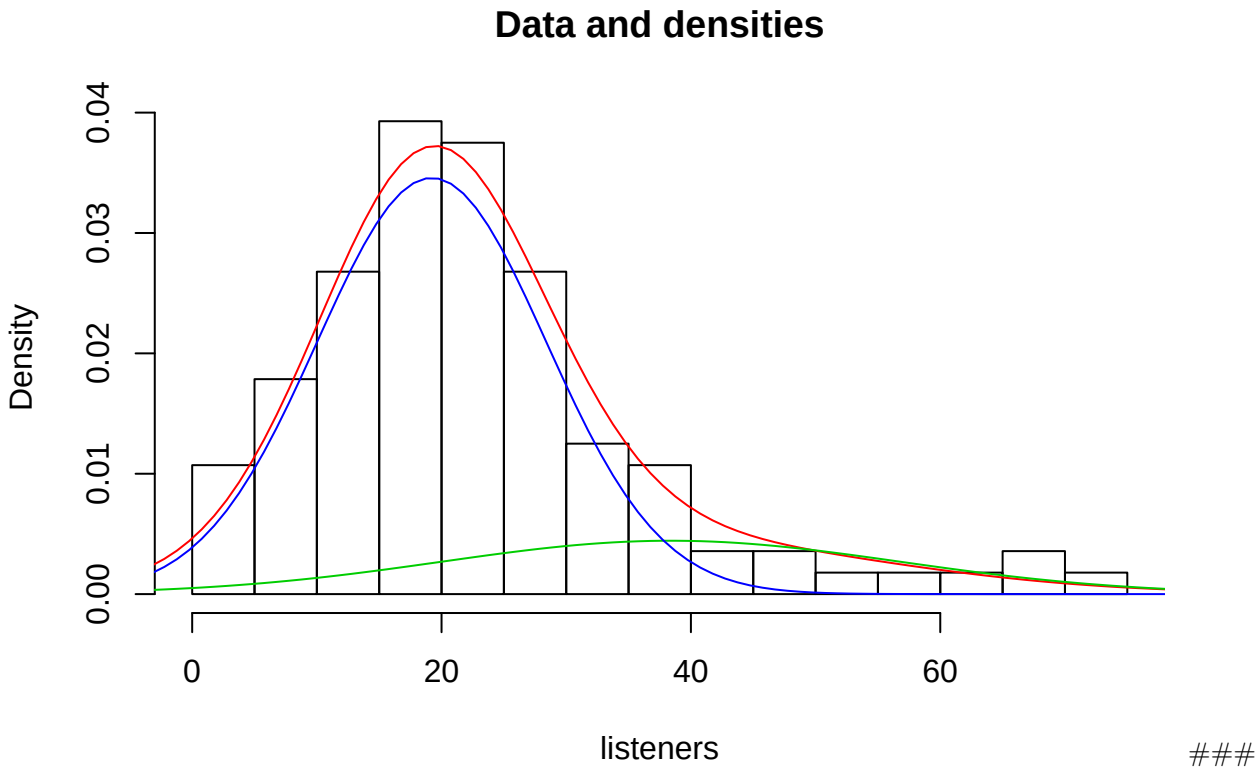


The chains

seem to reach stationarity rather quickly, in less than 50 iterations.

3b

```
piMean = mean(gibbs$piSample[51:nIter])
mu1Mean = mean(gibbs$muSample[51:nIter,1])
mu2Mean = mean(gibbs$muSample[51:nIter,2])
sigma21Mean = mean(gibbs$sigma2Sample[51:nIter,1])
sigma22Mean = mean(gibbs$sigma2Sample[51:nIter,2])

hist(listeners,n=20,freq=F,main="Data and densities")
lines(gibbs$mixDensGrid,gibbs$mixDensMean,col=2)
lines(gibbs$mixDensGrid,piMean*dnorm(gibbs$mixDensGrid,mu1Mean,sqrt(sigma21Mean)),col=3)
lines(gibbs$mixDensGrid,(1-piMean)*dnorm(gibbs$mixDensGrid,mu2Mean,sqrt(sigma22Mean)),col=4)
```



###

3b

```
print(paste("mu_1 mean: ",mu1Mean))
```

```
## [1] "mu_1 mean:  38.3908367119697"
```

```
print(paste("mu_2 mean: ",mu2Mean))
```

```
## [1] "mu_2 mean:  19.2115169219987"
```

```
print(piMean)
```

```
## [1] 0.2044454
```

Since $\mu_1 > \mu_2$, the first component corresponds to hit songs. Thus, the probability that the song becomes a hit song is the same as π, which has posterior mean 0.20.

Given that the song is a hit song means that its number of listeners x^* is distributed according to the first component $\phi(x^*|\mu_1, \sigma_1^2)$. The posterior predictive distribution is $p(x^*|x_1, \ldots, x_n) =$

$\int \phi(x^*|\mu_1, \sigma_1^2) p(\mu_1, \sigma_1^2|x_1, \ldots, x_n) d\mu_1 d\sigma_1^2$, which can be computed using simulation and the probability be computed using the samples.

```
predSim <- rnorm(nIter-50,gibbs$muSample[51:nIter,1],sqrt(gibbs$sigma2Sample[51:nIter,1]))
mean(predSim>60)
```

```
## [1] 0.1336842
```

The probability that a hit song gets more than 60 million listeners is 0.13.

Problem 4

4a

```
x <- zinc
n <- length(x)

# Log pdf for the truncated normal
logdtruncnorm <- function(x, mu, sigma, a){
  logdens <- dnorm((x-mu)/sigma,log=TRUE) - log(sigma) - log(1-pnorm((a-mu)/sigma))
  return(logdens)
}

logprior <- function(mu,sigma) {
  dnorm(mu,1000,100,log=TRUE) + dnorm(sigma,1000,100,log=TRUE)
}

# Define the posterior (likelihood x prior)
logpostTruncnorm <- function(param,x){
  mu = param[1]
  sigma = param[2]
  sum(logdtruncnorm(x, mu, sigma, a = 400)) + logprior(mu,sigma)
}

initVal = c(1000,1000)
optRes <- optim(par = initVal, fn  = logpostTruncnorm, gr = NULL, x, method = c("L-BFGS-B"), control = l
postMean <- optRes$par # This is the mean vector
postCov <- -solve(optRes$hessian) # This is posterior covariance matrix

print(postMean)
```

```
## [1] 1056.113 1002.717
```

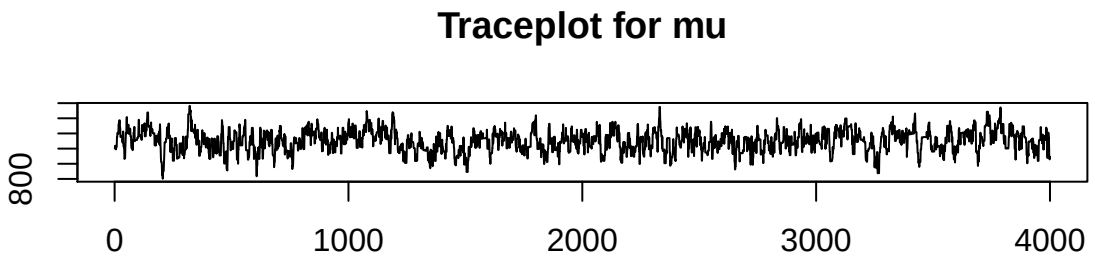
```
print(postCov)
```

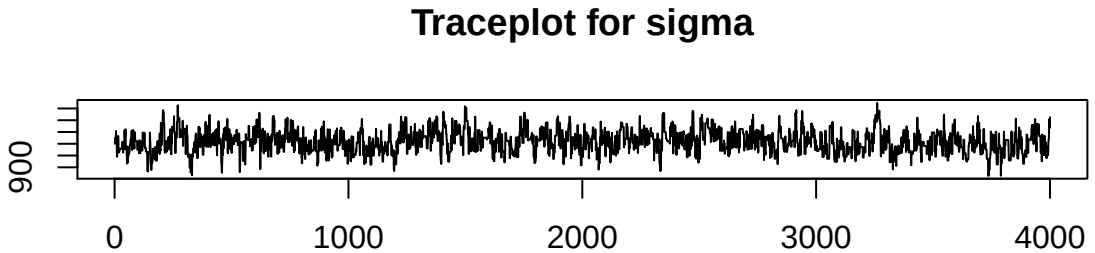
```
##           [,1]      [,2]
## [1,]  5683.082 -2465.765
## [2,] -2465.765  2522.061
```

4b

```
Metropolis <- function(c,niter,warmup,initVal,Sigma,logPostFunc,...) {

  theta <- initVal
  thetamat <- matrix(0,length(theta),warmup+niter)
  thetamat[,1] <- theta
```

```
  accprobvec <- rep(0,warmup+niter)

  for(i in 2:(warmup+niter)) {
    thetaProp <- runif(2,min = theta - 100, theta + 100)
    accprob <- exp(logPostFunc(thetaProp,...) - logPostFunc(theta,...))
    accprobvec[i] <- min(accprob,1)
    if(runif(1) < accprob) {
      theta <- thetaProp
    }
    thetamat[,i] <- theta
  }

  return(list(thetamat=thetamat,accprobvec=accprobvec))
}

c <- 5
niter <- 3000
warmup <- 1000

initVal <- c(1000,1000)

mp <- Metropolis(c,niter,warmup,initVal,Sigma,logpostTruncnorm,x)

par(mfrow=c(2,1))
plot(mp$thetamat[1,], type="l", main="Traceplot for mu",xlab="",ylab="")
plot(mp$thetamat[2,], type="l", main="Traceplot for sigma",xlab="",ylab="")
```





The traceplots shows fast variation which indicates that the sampler is efficient. However, the efficiency could probably be improved further with a better proposal distribution.

```
par(mfrow=c(2,1))
hist(mp$thetamat[1,(1:niter)+warmup], main="Posterior for mu",xlab="",ylab="",freq=F)
grid <- seq(500,1500,1)
```

```
lines(grid,dnorm(grid,postMean[1],sqrt(postCov[1,1])),col=2)

hist(mp$thetamat[2,(1:niter)+warmup], main="Posterior for sigma",xlab="",ylab="",freq=F)
grid <- seq(500,1500,1)
lines(grid,dnorm(grid,postMean[2],sqrt(postCov[2,2])),col=2)
```