Solution to computer exam in Bayesian learning

Per Sidén

2019-06-04

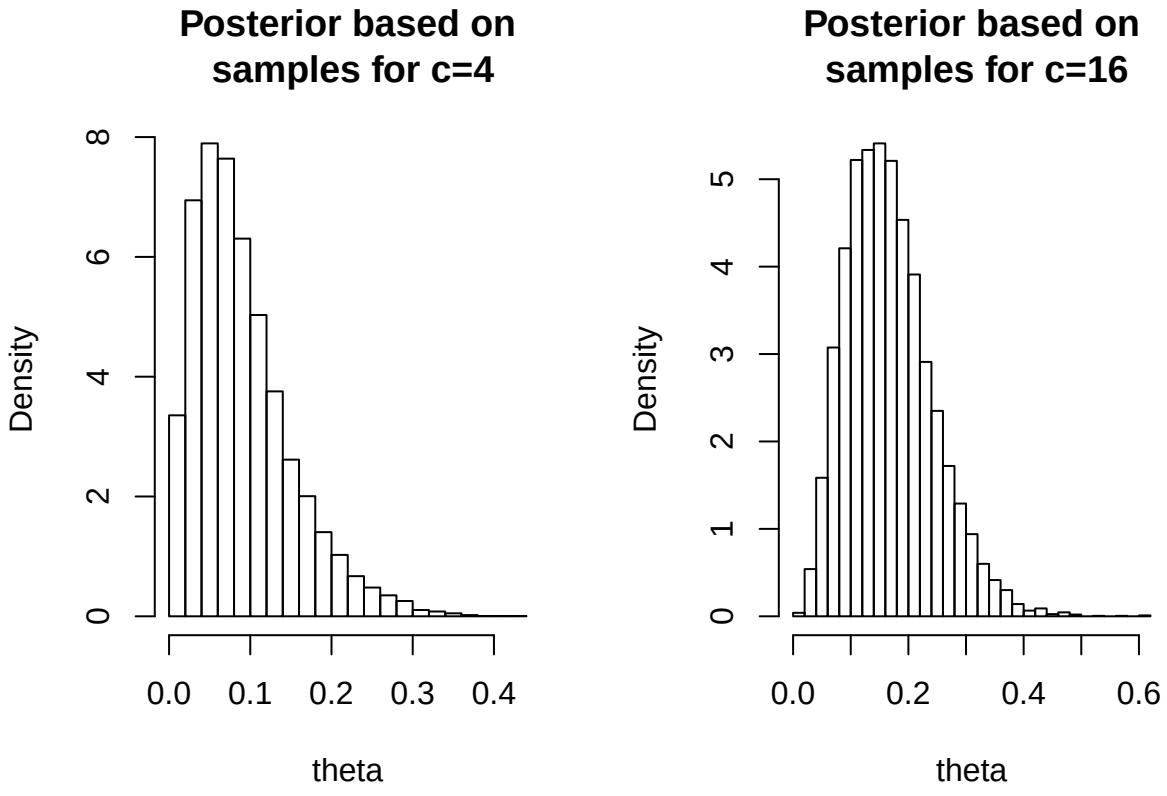
First load all the data into memory by running the R-file given at the exam

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

Problem 1

1a

```
nSamples = 10000 # 100000 #
beta = 20
postSamples1 = rbeta(nSamples,sqrt(4),beta)
postSamples2 = rbeta(nSamples,sqrt(16),beta)
par(mfrow=c(1,2))
hist(postSamples1,30,freq=F, main="Posterior based on\n samples for c=4",xlab="theta")
hist(postSamples2,30,freq=F, main="Posterior based on\n samples for c=16",xlab="theta")
```



1b

```
print(c(mean(postSamples1>.1),mean(postSamples2>.1)))
```

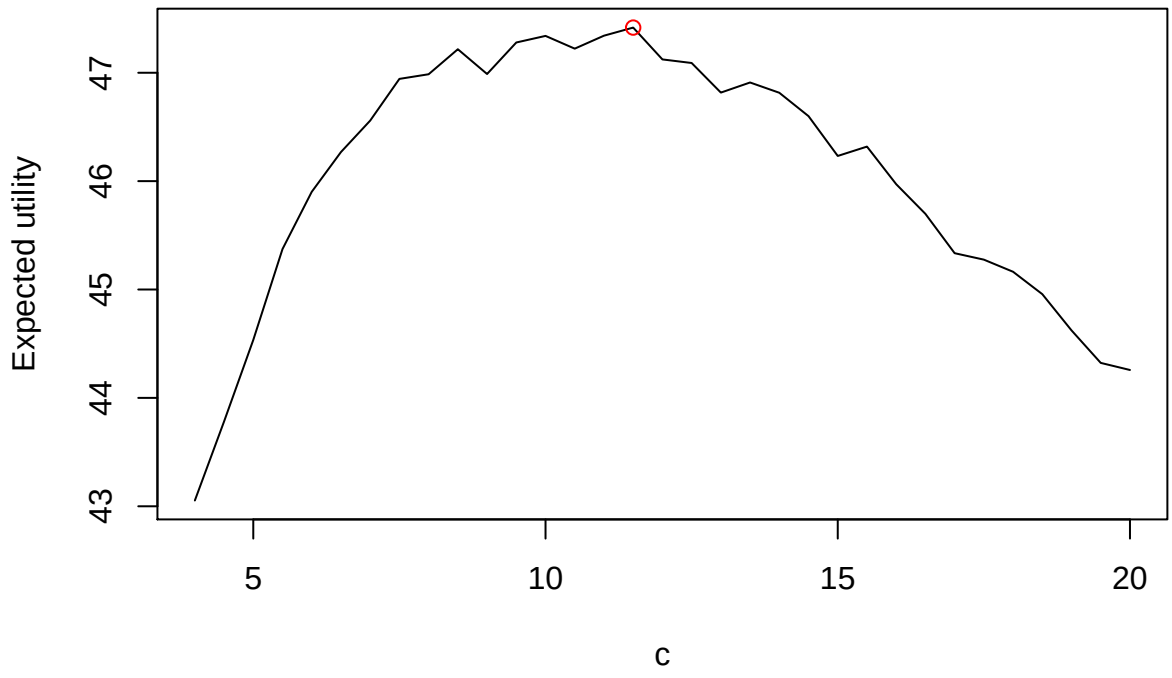
```
## [1] 0.3572 0.8110
```

Answer: The probabilities are roughly 0.36 and 0.81.

1c

Find the value of c that gives the maximum expected utility

```
cgrid = seq(4,20,.5)
utility <- function(theta,c){
  return(100+20*log(theta)-c)
}
expectedUtility <- function(c){
  postSamples = rbeta(nSamples,sqrt(c),beta)
  return(mean(sapply(postSamples,utility,c=c)))
}
EU = sapply(cgrid,expectedUtility)
cOpt = cgrid[which.max(EU)]
par(mfrow=c(1,1))
plot(cgrid,EU,type="l",ylab="Expected utility",xlab="c")
points(x=cOpt, y = max(EU), col = "red")
```




```
print(cOpt)
```

```
## [1] 11.5
```

Answer: The optimal value is roughly at $c = 10.5$ (but can be between 9 and 11.5 due to randomness).

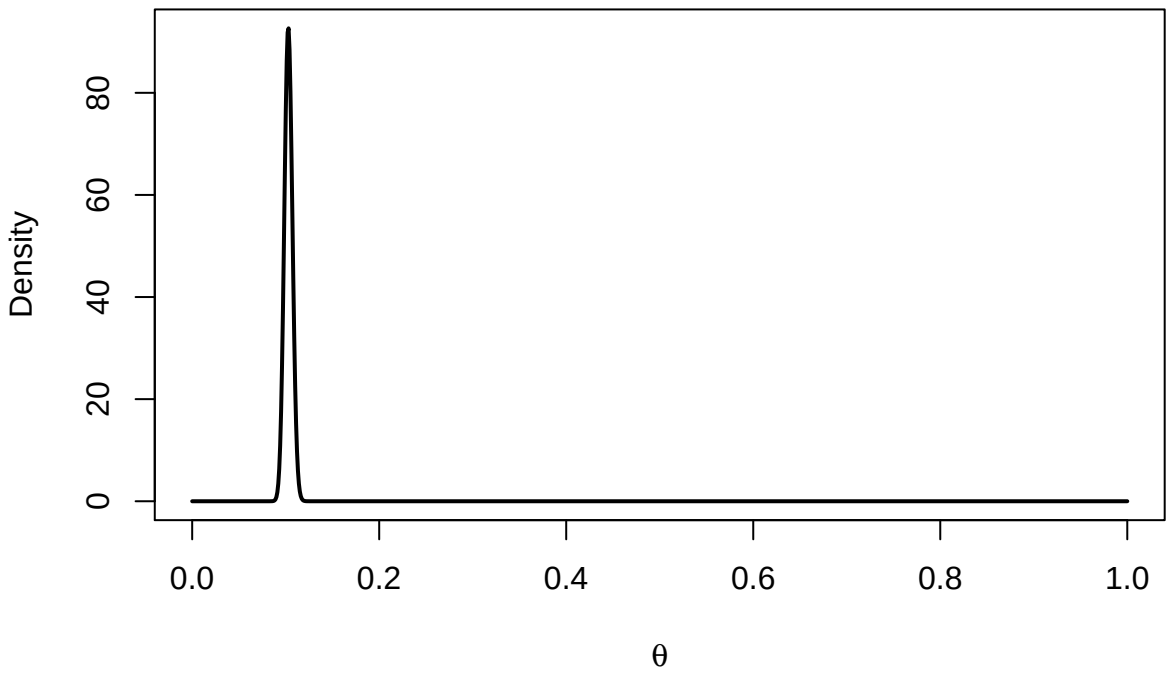
```
# # Incorrect solution: evaluate the utility at the mean, gives c=8.5
# thetaMean <- function(c){
```

```
#   return(sqrt(c)/(sqrt(c)+beta))
# }
#
# meanU = mapply(utility,thetaMean(cgrid),cgrid)
# cOptMean = cgrid[which.max(meanU)]
# plot(cgrid,meanU,type="l")
# points(x=cOptMean, y = max(meanU), col = "red")
# print(cOptMean)
```

Problem 2

2a

```
# Reading data
load(file = 'ebay.RData')

gridn = 1000
thetaGrid <- seq(0,1,length = gridn)
logPost <- function(theta,x){

  # evaluating the log-likelihood
  logLik <- sum(dbinom(x,size=50,prob=theta,log=T))
  # logLik <- sum(x*log(theta) + (50-x)*log(1-theta))

  # evaluating the prior
  logPrior <- 2*log(1-theta)

  # add the log prior and log-likelihood together to get log posterior
  logPost <- logLik + logPrior
  # if (abs(logPost) == Inf || is.na(logPost)) logPost = -20000;
  return(logPost)
}

logPostUnnorm <- vector(length=length(thetaGrid))
for(i in 1:length(thetaGrid)) logPostUnnorm[i] <- logPost(thetaGrid[i],ebay)
# logPostUnnorm = logPostUnnorm - max(logPostUnnorm) # For numerical stability
gridWidth = thetaGrid[2] - thetaGrid[1]
postGrid <- (1/gridWidth)*exp(logPostUnnorm)/sum(exp(logPostUnnorm))
plot(thetaGrid, postGrid, type = "l", lwd = 2, main="Posterior",
     ylab = "Density", xlab = expression(theta))
```




```
thetaMode = thetaGrid[which.max(postGrid)]
print(thetaMode)
```

```
## [1] 0.1031031
```

Answer: The mode is at $\theta = 0.103$.

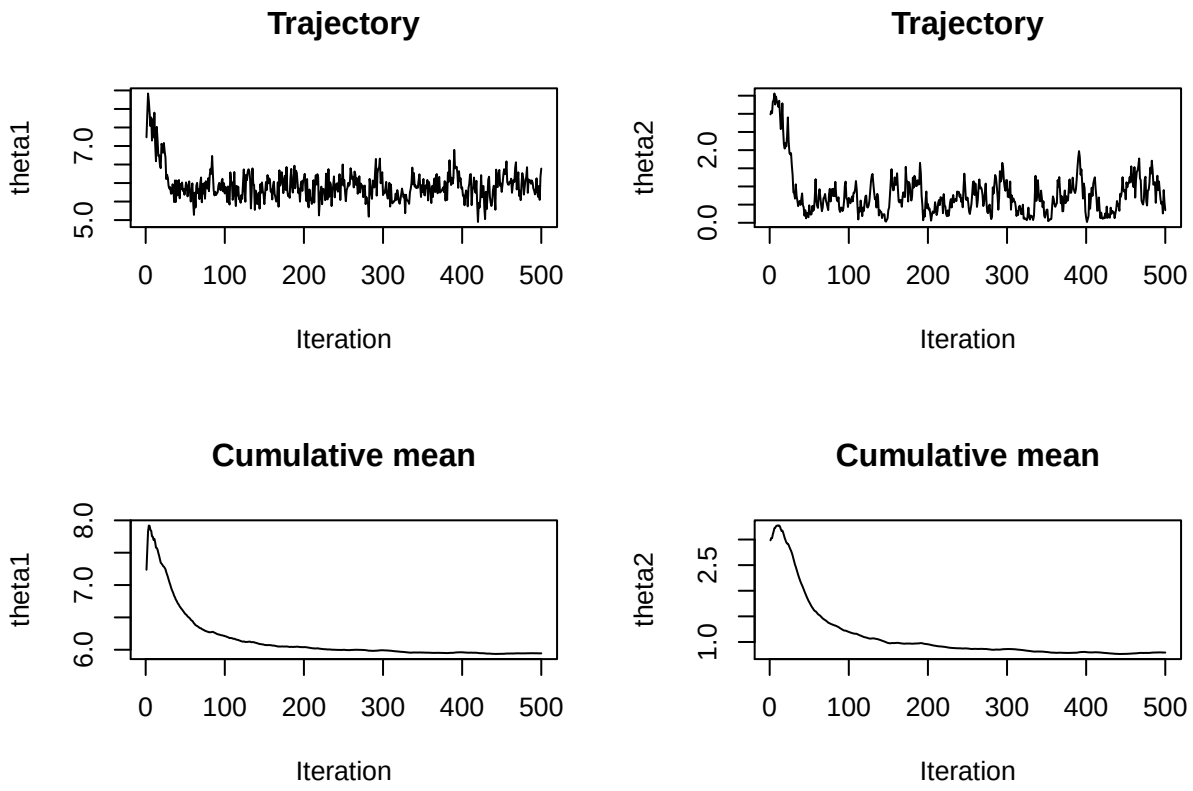
2b

We now use the GibbsMixPois function in the ExamData.R code to simulate with $K = 2$ (Hide iteration count output using message=FALSE)

```
set.seed(100)
GibbsResults2 <- GibbsMixPois(x = ebay, nComp = 2, alpha = 1, alphaGamma = 1, betaGamma = 1,
                              xGrid = xGrid, nIter = 500)
```

Plot trajectories and cumulative means

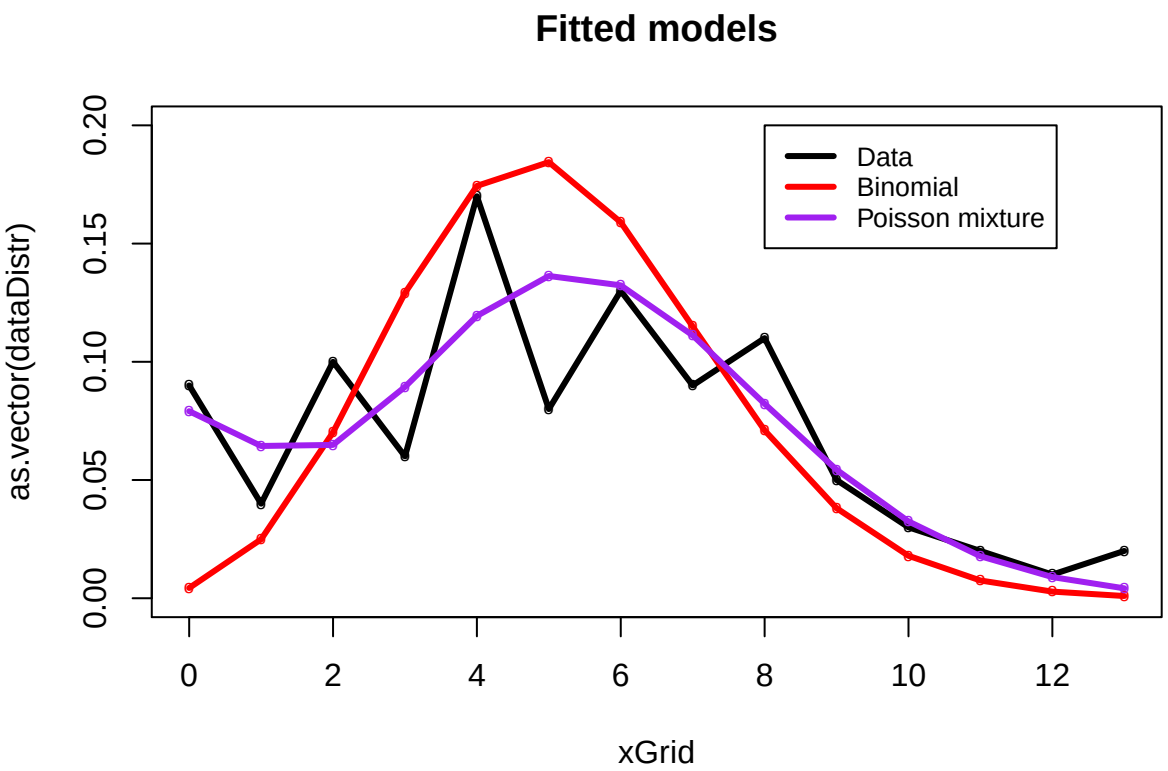
```
par(mfrow=c(2,2))
plot(GibbsResults2$thetaSample[,1],type="l",xlab="Iteration",ylab="theta1",main="Trajectory")
plot(GibbsResults2$thetaSample[,2],type="l",xlab="Iteration",ylab="theta2",main="Trajectory")
plot(cumsum(GibbsResults2$thetaSample[,1])/seq(1:500),type="l",xlab="Iteration",
     ylab="theta1",main="Cumulative mean")
plot(cumsum(GibbsResults2$thetaSample[,2])/seq(1:500),type="l",xlab="Iteration",
     ylab="theta2",main="Cumulative mean")
```



A burn-in of 50 iterations is sufficient for the chain to reach its stationary distribution, which can be seen in the trajectory plots.

2c

```
par(mfrow=c(1,1))
dataDistr = bidsCounts/sum(bidsCounts)

# thetaMode = .1 # If 2a not solved
binoDistr = dbinom(xGrid, size = 50, prob = thetaMode)
plot(xGrid, as.vector(dataDistr), type = "o", lwd = 3, col = "black", pch = 'o', cex = 0.6,
     ylim = c(0,0.2), main = "Fitted models")
lines(xGrid, binoDistr, type = "o", lwd = 3, col = "red", pch = 'o', cex = 0.6)
lines(xGrid, GibbsResults2$mixDensMean, type = "o", lwd = 3, col = "purple", pch = 'o', cex = 0.6)
legend(x = 8, y = 0.2, legend = c("Data", "Binomial","Poisson mixture"),
       col = c("black","red","purple"), lty = c(1,1), lwd = c(3,3), cex = 0.8)
```



The binomial model fits the data quite badly, which can mainly be seen in that the number of zero-observations is much higher in the dataset, than what the density suggests. The Poisson mixture gives a better fit.

Problem 3

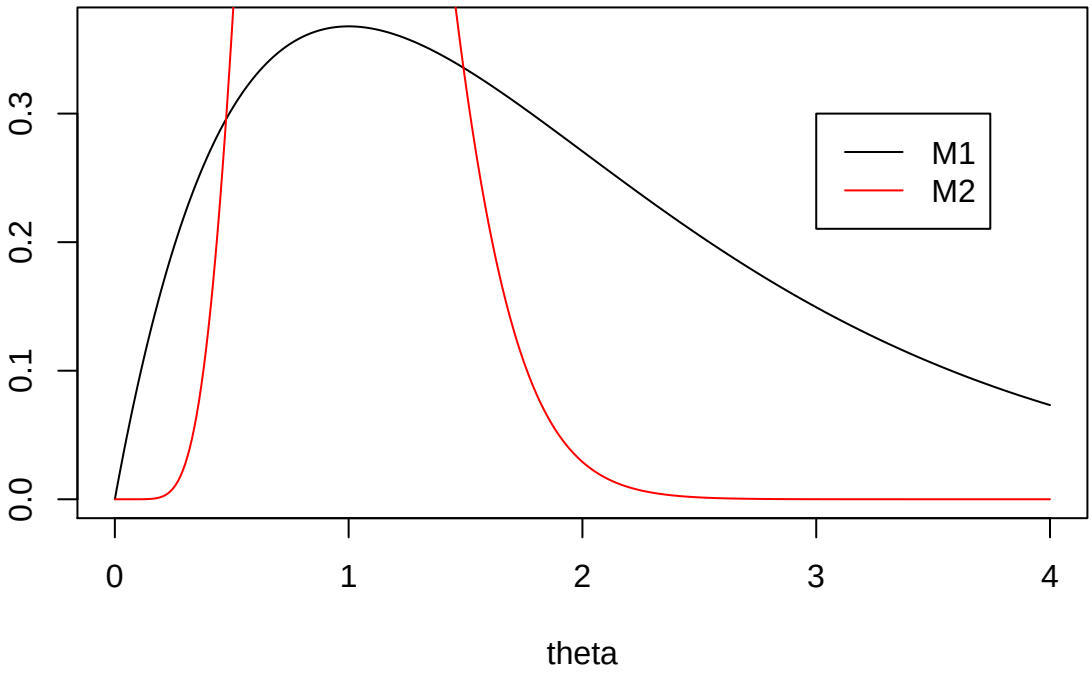
See solution on paper.

Problem 4

```
# Load cellphone lifetime data
load(file = 'cellphones.RData')
x = cellphones
```

4a

```
# hist(x)
n = length(x)
alpha1 = 2
beta1 = 1
alpha2 = 10
beta2 = 10

thetaGrid = seq(0,4,.01)
plot(thetaGrid,dgamma(thetaGrid,alpha1,beta1),type="l",ylab="",xlab="theta",main="Gamma prior densities
lines(thetaGrid,dgamma(thetaGrid,alpha2,beta2),col="red")
legend(x=3,y=.3,c("M1","M2"),col = c("black","red"), lty = c(1,1))
```



The M_1 prior has mean $\frac{2}{1} = 2$ and variance $\frac{2}{1 \cdot 1} = 2$. The M_2 prior has mean $\frac{10}{10} = 1$ and variance $\frac{10}{10 \cdot 10} = 0.1$. Comparing the variances tells us that the M_2 prior is more informative and this can also be seen by plotting the densities. This is because the M_2 prior is more concentrated and has thinner tails.

4b

```
logprior <- function(theta,alpha,beta){
  return(dgamma(theta,shape=alpha,rate=beta,log=T))
}
loglik <- function(x,theta){
  return(sum(dexp(x,theta,log=T)))
}
logposterior <- function(theta,alpha,beta,x){
  return(dgamma(theta,shape=alpha+length(x),rate=beta+sum(x),log=T))
}
logmarglik <- function(theta,alpha,beta,x){
  return(loglik(x,theta) + logprior(theta,alpha,beta) - logposterior(theta,alpha,beta,x))
}

lms = c(logmarglik(1,alpha1,beta1,x),logmarglik(1,alpha2,beta2,x))
unnormProbs = .5*exp(lms)
probs = unnormProbs/sum(unnormProbs)
print(probs)
```

```
## [1] 0.616943 0.383057
```

The posterior probability of M_1 is 0.617 so this is more probable, however the data does not strongly suggest that any of the models is better than the other.

4c

The interval can be computed in several ways, but using simulation is simple

```
ndraws = 100000
xTildeDraws = rep(0,ndraws)
for(i in 1:ndraws){
  M = rbinom(1,1,probs[2]) + 1 # Simulate which model to use
  if(M==1){
    theta = rgamma(1,shape=alpha1+length(x),rate=beta1+sum(x))
  } else {
    theta = rgamma(1,shape=alpha2+length(x),rate=beta2+sum(x))
  }
  xTildeDraws[i] = rexp(1,theta)
}

print(quantile(xTildeDraws,probs = c(.05,.95)))
```

```
##        5%       95%
## 0.1201326 7.4266830
```

The interval is roughly (0.12,7.5).