CGS698C, Assignment 03

Himanshu Yadav

2024-06-21

Part 1: Estimating the posterior distribution using different computational methods

You are given 10 independent and identically distributed datapoints that are assumed to come from a Binomial distribution with sample size 20 and probability of success θ:

10, 15, 15, 14, 14, 14, 13, 11, 12, 16

Model:

Let y_i be the i^{th} datapoint,

$$y_i \sim Binomial(n = 20, \theta)$$

$$\theta \sim Beta(1, 1)$$

The analytically-derived posterior distribution of θ is a Beta distribution with shape parameters 135 and 67.

$$\theta|y \sim Beta(135, 67)$$

Use the above information to solve the following exercises.

1. Graph the analytically-derived posterior distribution of θ.
2. Use **grid approximation** to estimate the posterior density function of θ and graph the estimated posterior density function.
3. Use Monte Carlo integration to estimate the marginal likelihood of the model. (Hint: Draw a large number of samples from the prior `Beta(1,1)`, compute the likelihood for each sample, and calculate the average of all the likelihoods.)
4. Use **importance sampling** to draw samples from the posterior distribution of θ.

   (Hint: Draw N samples of θ from a proposal density function, say $q(\theta)$. Compute the likelihood $L(\theta_i|y)$, prior $p(\theta_i)$, and proposal density $q(\theta_i)$ for each sample. Store each sample θ_i and its corresponding weight w_i as a row in a dataframe, where $w_i = \frac{L(theta_i|y)p(\theta_i)}{q(\theta_i)}$. Now, select $N/4$ samples from the vector of N samples based on their weights. You can use the function $sample(theta\_vector, size = n/4, prob = weights\_vector)$. These new $N/4$ samples are the samples from the posterior.)

```
# Assuming df is the dataframe containing samples of theta and
# their correponding weights as the columns 'theta' and 'weights' respectively.
```

```
n <- length(df$theta)
sample(df$theta, size=n/4,prob=df$weights)
```

5. Use Markov chain Monte Carlo method to estimate the posterior distribution of θ.

   (Hint: You can use the Metropolis-Hastings algorithm code given on page 20 in the lecture notes.)

6. Graphically compare the posterior distributions of θ obtained using:

   - Importance sampling
   - Markov chain Monte Carlo
   - Analytical derivation

Part 2: Writing your own sampler for Bayesian inference

2.1 The research problem

In a visual word recognition experiment, a participant has to recognize whether a string shown on the screen is a meaningful word (e.g., "book") or a non-word (e.g., "bktr"). The participant is asked to answer "yes" if the shown string is a meaningful word, and "no" if it is a meaningless non-word. Suppose a participant is shown n words and n non-words on the screen one by one and you record the recognition time for each word/non-word.

Say, T_w is the vector of word recognition times, and T_{nw} is the vector of non-word recognition times.

You ask the following question:

Does it take longer to recognize the non-words compared to the words?

Technically,

Is the mean recognition time for the non-words larger than the mean recognition time for the words?

2.2 Hypothesis

Lexical-access hypothesis: The mean recognition time for the non-words is longer than the mean recognition time for the non-words.

2.3 Model

RT_i is the observed recognition time of the i^{th} string, which can be a word or a non-word.

Likelihood assumption:

$$RT_i \sim Normal(\mu_i, \sigma)$$

such that

$$\mu_i = \alpha + \beta \cdot type_i$$

where $type_i$ indicate whether the i^{th} string is a word or a non-word. If the string i is a non-word, the $type_i$ will be equal to 1, otherwise 0. α, β, and σ are the three parameters of the model; α represents mean recognition time for the words, and β represents to what extent non-words are recognized slower than words.

Priors:

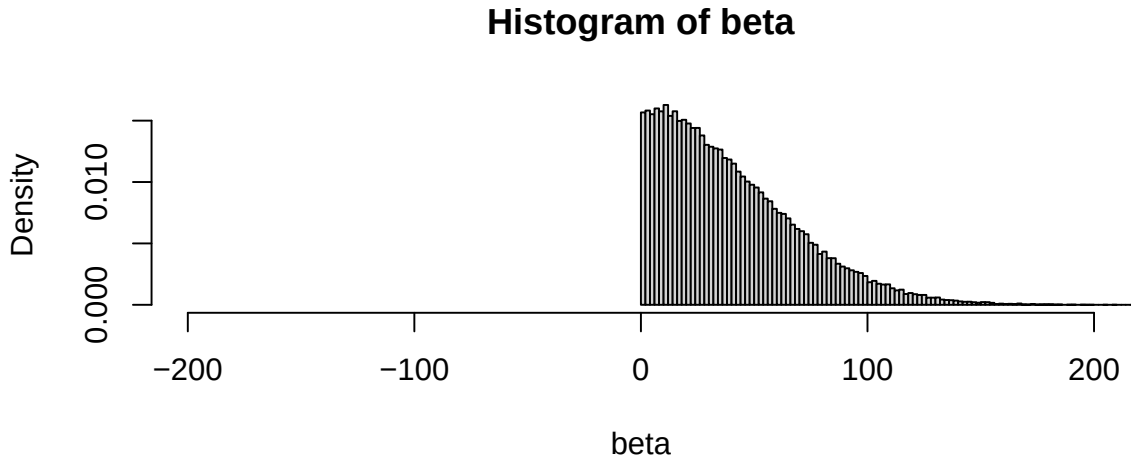
$$\alpha \sim Normal(400, 50)$$

$$\sigma = 30$$

$$\beta \sim Normal_+(0, 50)$$

where $Normal_+()$ represent a truncated normal distribution such that β would always be larger than 0.

```
library(truncnorm)
# You can generate from a truncated normal distribution using rtruncnorm
beta <- rtruncnorm(100000,a=0,b=Inf,mean=0,sd=50)
hist(beta,xlim = c(-200,200),probability = T,breaks = 100)
```




```
# You can calculate the probability density of obtaining a value x
# from the truncated normal distribution.
x <- 20
density_x <- dtruncnorm(x,a=0,b=Inf,mean=0,sd=50)
```

2.4 Data

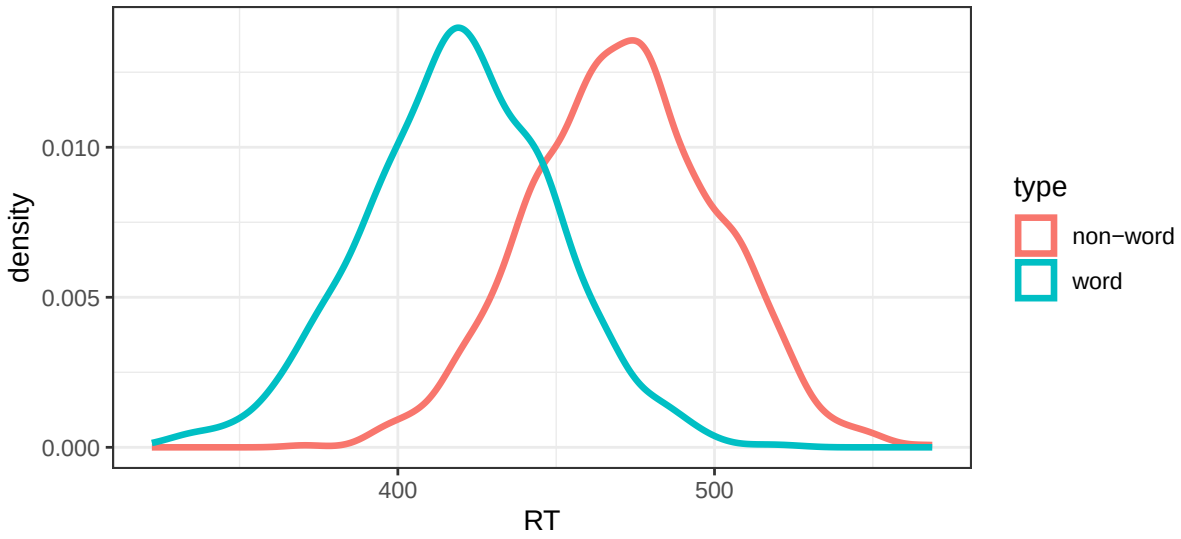
The file `word-recognition-times.csv` on github contains the recognition times data for 4000 strings which can be either words or non-words represented by the column *type*; the recognition times are given in the column *RT*. Use the following code to load the data. (Alternatively, you can directly download the csv file and read it using `read.table`.)

```
dat <- read.table(
  "https://raw.githubusercontent.com/yadavhimanshu059/CGS698C/main/notes/Data/word-recognition-ti
  sep=",",header = T)[,-1]
head(dat)
```

```
##       type       RT
## 1     word 423.1019
## 2     word 429.9432
## 3 non-word 486.9959
## 4 non-word 451.4400
## 5 non-word 482.2657
## 6 non-word 470.8003
```

```
ggplot(dat,aes(x=RT,color=type))+geom_density(size=1.2)+theme_bw()
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



2.5 Exercises

2.5.1 Estimate the parameters α and β using **Markov Chain Monte Carlo.**

(Hint: Transform posterior densities on log scale. See the algorithm on page 24 in the lecture notes.)

2.5.2 What is the 95% credible interval for each parameter?

(If you can say with 95% certainty that the true value of the parameter lies between a and b, then the range $[a, b]$ is called the credible interval. You can estimate the credible interval from the vector of samples from the posterior, using `quantile(post_samples,probs=c(.025,.975))`)

Part 3: Hamiltonian Monte Carlo sampler

You are given 500 data points that are assumed to come from a normal distribution with mean μ and variance σ^2,

```
true_mu <- 800
true_var <- 100 #sigma^2
y <- rnorm(500,mean=true_mu,sd=sqrt(true_var))
hist(y)
```

Copy and paste the above chunk of code to generate data y.
Model:
Let y_i be i^{th} data point,
$y_i \sim Normal(\mu, \sigma^2)$
$\mu \sim Normal(m = 1000, s = 100)$
$\sigma \sim Normal(a = 10, b = 2)$
You have to estimate the posterior distributions of μ and σ using Hamiltonian Monte Carlo sampler.

You can directly use the following HMC sampler code to estimate the parameters of the above model.

```
#Gradient functions
gradient <- function(mu,sigma,y,n,m,s,a,b){
  grad_mu <- (((n*mu)-sum(y))/(sigma^2))+((mu-m)/(s^2))
  grad_sigma <- (n/sigma)-(sum((y-mu)^2)/(sigma^3))+((sigma-a)/(b^2))
  return(c(grad_mu,grad_sigma))
}
#Potential energy function
V <- function(mu,sigma,y,n,m,s,a,b){
  nlpd <- -(sum(dnorm(y,mu,sigma,log=T))+dnorm(mu,m,s,log=T)+dnorm(sigma,a,b,log=T))
  nlpd
}
```

```r
#HMC sampler
HMC <- function(y,n,m,s,a,b,step,L,initial_q,nsamp,nburn){
  mu_chain <- rep(NA,nsamp)
  sigma_chain <- rep(NA,nsamp)
  reject <- 0
  #Initialization of Markov chain
  mu_chain[1] <- initial_q[1]
  sigma_chain[1] <- initial_q[2]
  #Evolution of Markov chain
  i <- 1
  while(i < nsamp){
    q <- c(mu_chain[i],sigma_chain[i])   # Current position of the particle
    p <- rnorm(length(q),0,1)            # Generate random momentum at the current position
    current_q <- q
    current_p <- p
    current_V = V(current_q[1],current_q[2],y,n,m,s,a,b) # Current potential energy
    current_T = sum(current_p^2)/2                       # Current kinetic energy
    # Take L leapfrog steps
    for(l in 1:L){
      # Change in momentum in 'step/2' time
      p <- p-((step/2)*gradient(q[1],q[2],y,n,m,s,a,b))
      # Change in position in 'step' time
      q <- q + step*p
      # Change in momentum in 'step/2' time
      p <- p-((step/2)*gradient(q[1],q[2],y,n,m,s,a,b))
    }
    proposed_q <- q
    proposed_p <- p
    proposed_V = V(proposed_q[1],proposed_q[2],y,n,m,s,a,b)  # Proposed potential energy
    proposed_T = sum(proposed_p^2)/2                         # Proposed kinetic energy
    accept.prob <- min(1,exp(current_V+current_T-proposed_V-proposed_T))
    # Accept/reject the proposed position q
    if(accept.prob>runif(1,0,1)){
      mu_chain[i+1] <- proposed_q[1]
      sigma_chain[i+1] <- proposed_q[2]
      i <- i+1
    }else{
      reject <- reject+1
    }
  }
```

```
  posteriors <- data.frame(mu_chain,sigma_chain)[-(1:nburn),]
  posteriors$sample_id <- 1:nrow(posteriors)
  posteriors
}
```

Exercise 3.1 Use the following values for the internal parameters of the HMC sampler,

- Total number of samples, $nsamp = 6000$
- Total number of burn-in samples, $nburn = 2000$ (a certain number of initial samples that you want to throw away)
- Step-size parameter, $step = 0.02$
- Number of leapfrog steps, $L = 12$
- Initializing value of the *mu* and *sigma* chain, $initial_q = c(1000, 11)$

Estimate and plot the posteriors for μ and σ.
(Hint: You can directly run the HMC function as follows:)

```
df.posterior <- HMC(y=y,n=length(y),          # data
                    m=1000,s=20,a=10,b=2,     # priors
                    step=0.02,                # step-size
                    L=12,                     # no. of leapfrog steps
                    initial_q=c(1000,11),      # Chain initialization
                    nsamp=6000,               # total number of samples
                    nburn=2000)               # number of burn-in samples
```

Exercise 3.2 Check posterior sensitivity to the total number of samples. How do the posteriors change with change in total number of samples?

Estimate and compare the posteriors obtained for the following values of total number of samples, $nsamp$ -

- $nsamp = 100$
- $nsamp = 1000$
- $nsamp = 6000$

Change burn-in samples $nburn$ accordingly. For example, you can set $nburn = nsamp/3$. Keep all other parameters same as in Expercise 1.1.

Exercise 3.3 How do the posteriors change with change in step-size parameter?

Estimate and compare the posteriors obtained when step-size had the ollowing values -

- $step = 0.001$
- $step = 0.005$

- $step = 0.02$

Keep all other parameter same as in Exercise 1.1.

Exercise 3.4 Visually inspect the *mu* and *sigma* chains obtained in expercise 3.3. Do you find anything problematic?

Exercise 3.5 Check the prior sensitivity for the μ parameter. Estimate and compare the posterior distribution of μ when the prior on μ are -

- $\mu \sim Normal(m = 400, s = 5)$
- $\mu \sim Normal(m = 400, s = 20)$
- $\mu \sim Normal(m = 1000, s = 5)$
- $\mu \sim Normal(m = 1000, s = 20)$
- $\mu \sim Normal(m = 1000, s = 100)$

(Keep all other parameters same as in Exercise 3.1).