Week 5 lab solutions – MAST90125: Bayesian Statistical learning

Question One

Consider a Poisson regression,

$$y_i \sim \text{Pois}(\lambda_i) \quad \text{and} \quad \log(\lambda_i) = \mathbf{x}_i'\boldsymbol{\beta}, \quad \boldsymbol{\beta} \in \mathbb{R}^p$$

In lectures we learned various techniques for approximating the posterior distribution. In this lab, attempt as many of these techniques as possible to complete the following tasks.

Consider the dataset `Warpbreaks.csv`, which can be downloaded from Canvas. This dataset contains information of the number of breaks in a consignment of wool. In addition, Wool type (A or B) and tension level (L, M or H) are recorded. To investigate the association between the number of breaks and wool type and tension level, various forms of generalised linear model are proposed where Bayesian computing techniques should be used.

As a reminder the techniques we considered were:

- Direct approximation.
- Rejection sampling.
- Importance sampling (with re-sampling).

When coding, assume the prior for the coefficients $\boldsymbol{\beta} \sim N(\mathbf{0}, 5\mathbf{I}_p)$.

Some hints:

An initial guess can be determined from fitting a Poisson regression using the function `glm`. Treat wool type as a factor using the function `glm`

```
warpbreak= read.csv(file = './warpbreaks.csv',header=TRUE)
    #This line will need to be changed when you run this yourself.
mod<-glm(breaks~as.factor(wool),data=warpbreak,family='poisson')
summary(mod)
```

```
##
## Call:
## glm(formula = breaks ~ as.factor(wool), family = "poisson", data = warpbreak)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       3.43518    0.03454  99.443  < 2e-16 ***
## as.factor(wool)B -0.20599    0.05157  -3.994 6.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 297.37  on 53  degrees of freedom
## Residual deviance: 281.33  on 52  degrees of freedom
## AIC: 560
##
## Number of Fisher Scoring iterations: 4
```

```
Sigma <-vcov(mod); Sigma
```

```
##                   (Intercept) as.factor(wool)B
## (Intercept)       0.001193293     -0.001193293
## as.factor(wool)B -0.001193293      0.002659566
```

```
X<-model.matrix(mod)
```

Direct approximation code.

To make things easier, we will only investigate the effect of wool.

```
#Step one: Build grid for parameters with N = 1000
N<-0:999
beta<-matrix(N,2,1000,byrow=TRUE)
 #Creating a range of beta values that cover the beta's domain:
for(i in 1:2){
beta[i,]<-mod$coef[i]-10*sqrt(Sigma[i,i]) + beta[i,]*20*sqrt(Sigma[i,i])/1000
}
n<-dim(X)[1] #n=54 data points
  #matrix of priors:
prior<-dnorm(beta[1,],mean=0,sd=sqrt(5))%*%t(dnorm(beta[2,],mean=0,sd=sqrt(5)))
 #Building matrix of candidate lambda values for each individual
 #and calculate unnormalised density.
post.norm<-prior # counter for un-normalised density.
for(i in 1:n){
  #N by N matrix of lambda values at (beta0, beta1) for observation i.
pc <- outer(X=beta[1,], Y=beta[2,], FUN = function(x1,y1) exp(x1+y1*X[i,2]))
  #likelihood for observation i at all possible values of \lambda (in 2-D grid)
  #multiplied by likelihood of previously evaluated observations:
post.norm<-dpois(warpbreak$breaks[i],lambda=pc)*post.norm
  #Adding observation i onto joint distribution y_1, ... y_{i-1}
}

#normalise discretised density.
post.discrete <-post.norm/sum(post.norm)

#Calculate posterior means.
#intercept. Note rowSums because beta0 was X in outer function.
beta0m<-sum(beta[1,]*rowSums(post.discrete));beta0m
```

```
## [1] 3.433715
```

```
#slope. Note colSums because beta1 was Y in outer function.
beta1m<-sum(beta[2,]*colSums(post.discrete));beta1m
```

```
## [1] -0.2051955
```

```
#Calculate posterior standard deviations.
#intercept. Note rowSums because beta0 was X in outer function.
beta0sd<-sqrt(sum(beta[1,]^2*rowSums(post.discrete))-beta0m^2);beta0sd
```

```
## [1] 0.03456149
```

```
#slope. Note colSums because beta1 was Y in outer function.
beta1sd<-sqrt(sum(beta[2,]^2*colSums(post.discrete))-beta1m^2);beta1sd
```

```
## [1] 0.05158127
```

```r
#95 % credible intervals

#function for finding quantiles for central credible interval.
inds<-function(x,alpha){
a<-max(which(cumsum(x)<0.5*alpha))
b<-min(which(cumsum(x)>1-0.5*alpha))
return(c(a,b))
}

#Intercept, 95 % central credible interal
beta[1,inds(rowSums(post.discrete),0.05) ]
```

```
## [1] 3.364711 3.500815
```

```r
#Slope, 95 % central credible interval
beta[2,inds(colSums(post.discrete),0.05)]
```

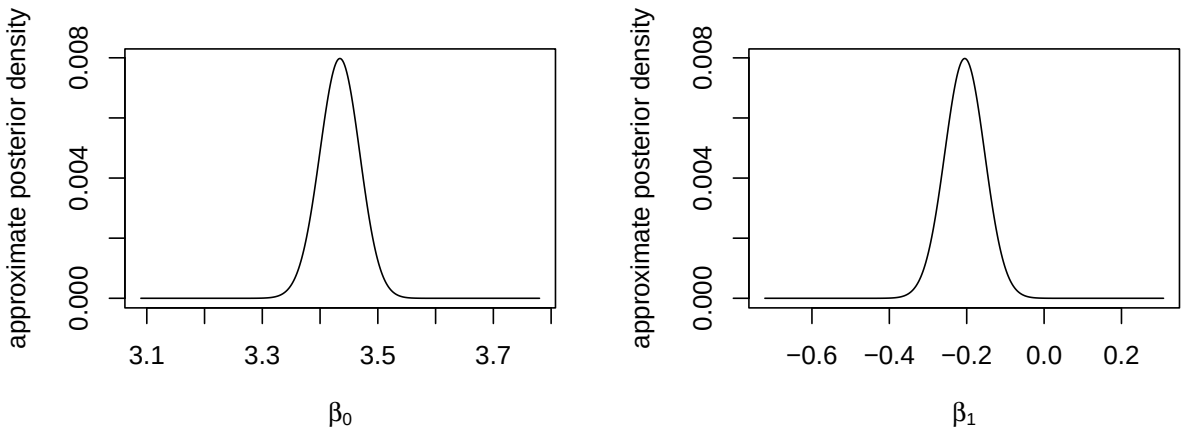
```
## [1] -0.3070676 -0.1038779
```

```r
#Compare to glm 95 % CI.
confint(mod,level=0.95)
```

```
## Waiting for profiling to be done...
```

```
##                       2.5 %     97.5 %
## (Intercept)       3.366701  3.5021304
## as.factor(wool)B -0.307263 -0.1050641
```

```r
par(mfrow=c(1,2))
#Plot marginal posteriors
plot(beta[1,],rowSums(post.discrete),pch=19,type='l',
     xlab=expression(beta[0]),ylab='approximate posterior density')
plot(beta[2,],colSums(post.discrete),pch=19,type='l',
     xlab=expression(beta[1]),ylab='approximate posterior density')
```



Rejection sampling code

```
#Let beta0, beta1 be drawn from
#uniform U(3.1,3.6) and uniform U(-0.4,0.2) respectively
#the estimate maximum of likelihood is estimated using glm, and the prior is maximised
#at beta_0=beta_1=0. as g(beta0) =1/(0.5) and g(beta1) =1/0.6 and are independent.
M<-prod(dpois(warpbreak$breaks,
  lambda=exp(X%*%mod$coefficients)))*dnorm(0,0,sqrt(5))*dnorm(0,0,sqrt(5))*0.5*0.6

#Drawing candidates. (Candidates are either accepted or rejected together)
iter       <- 100000 #number of iterations.
beta0.draw <-rep(NA,iter)
beta1.draw <-rep(NA,iter)
for(i in 1:iter){
   beta0.c <- runif(1,3.1,3.6)    #candidate intercept
   beta1.c <- runif(1,-0.4,0.2)   #candidate slope
   p.c     <- exp(beta0.c+beta1.c*X[,2]) #resulting candidate lambda.
   prob.a  <- prod(dpois(warpbreak$breaks,
      lambda=p.c))*dnorm(beta0.c,0,sqrt(5))*dnorm(beta1.c,0,sqrt(5))/(M/0.5/0.6)
   #acceptance probability.
   ind     <- rbinom(1,1,prob=prob.a) #accept candidate draw.
   beta0.draw[i] <- ind*beta0.c       #Substitute zero if ind = 0, candidate is ind is 1.
   beta1.draw[i] <- ind*beta1.c       #Substitute zero if ind = 0, candidate is ind is 1.
}
#remove rejects
beta0.draw<-beta0.draw[beta0.draw != 0]
beta1.draw<-beta1.draw[beta1.draw != 0]

length(beta0.draw)/iter #acceptance rate
```

```
## [1] 0.00847
```

```
#Posterior means
mean(beta0.draw); mean(beta1.draw)
```

```
## [1] 3.434573
```

```
## [1] -0.2073321
```

```
#Posterior standard deviations
sd(beta0.draw); sd(beta1.draw)
```

```
## [1] 0.03399375
```

```
## [1] 0.05123844
```

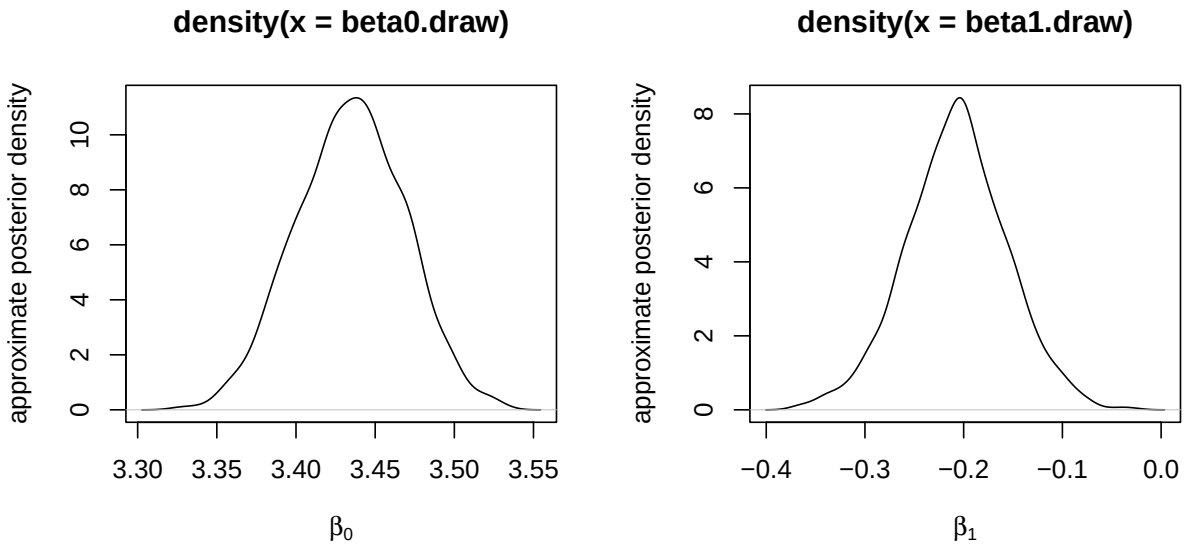
```
#95 % central credible intervals
quantile(beta0.draw,c(0.025,0.975))
```

```
##     2.5%    97.5%
## 3.366513 3.499342
```

```
quantile(beta1.draw,c(0.025,0.975))
```

```
##      2.5%     97.5%
## -0.311503 -0.104460
```

```
par(mfrow=c(1,2))
#Plot marginal posteriors
plot(density(beta0.draw),type='l',xlab=expression(beta[0]),ylab='approximate posterior density')
plot(density(beta1.draw),type='l',xlab=expression(beta[1]),ylab='approximate posterior density')
```



Importance sampling code

For importance sampling. Let's approximate a Poisson regression by fitting a lm to log-transformed responses.

```
modest  <-lm(log(breaks)~as.factor(wool),data=warpbreak)
 #fit lm with estimated log as response.

#Importance sampling distribution is normal with mean = betahat and
#variance-covariance matrix extracted from lm object.
betaest<-modest$coef
sigma  <-vcov(modest)

iter = 100000 #iterations.
beta.keep<-matrix(0,iter,2)     #Matrix of storing simulated values
w.all    <-rep(0,iter)          #vector to store importance weights.
library(mvtnorm)                #In order to draw from mv normal.
```

```
## Warning: package 'mvtnorm' was built under R version 4.3.1
```

```
for(i in 1:iter){
beta.c  <-rmvnorm(1,mean=betaest,sigma=sigma)   #draw candidate
beta.c<-as.numeric(beta.c)
```

```
beta.keep[i,]<-as.numeric(beta.c)     #turning random number to class numeric and store.
#calculating candidate lambda.
p.c     <- exp(X%*%beta.c)
 #Calculating the importance weight.
w.c     <- prod(dpois(warpbreak$breaks,lambda=p.c))*
  prod(dnorm(beta.c,0,sqrt(5)))/dmvnorm(beta.c,mean=betaest,sigma=sigma)
w.all[i]<-w.c   #store the importance weight.
}

#re-sample 5000 iterations without replacement.
ind.sam <-sample(iter,size=5000,prob=w.all,replace=FALSE) #find indices.
beta.resample<-beta.keep[ind.sam,]

#Posterior means
colMeans(beta.resample)
```

```
## [1]  3.430699 -0.204610
```

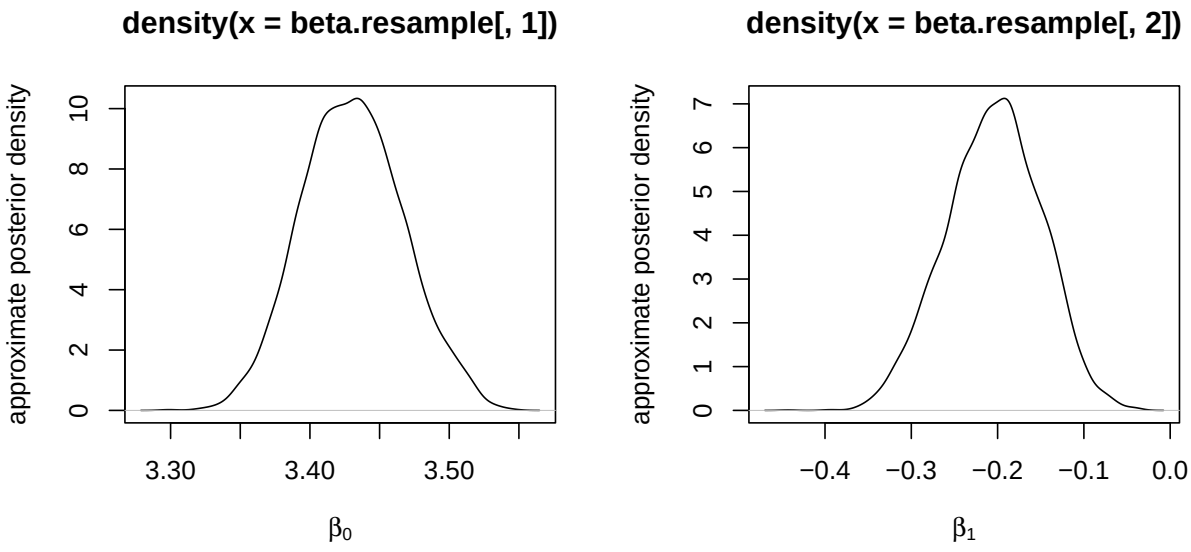
```
#Posterior standard deviations
apply(beta.resample,2,FUN=sd)
```

```
## [1] 0.03644599 0.05489943
```

```
#95 % central credible intervals
apply(beta.resample,2,FUN=function(x) quantile(x,c(0.025,0.975)) )
```

```
##           [,1]       [,2]
## 2.5%  3.361741 -0.3149976
## 97.5% 3.504309 -0.1021177
```

```
#Plots of approximate posterior densities from importance re-sampling.
par(mfrow=c(1,2))
#Plot marginal posteriors
plot(density(beta.resample[,1]),type='l',xlab=expression(beta[0]),ylab='approximate posterior density')
plot(density(beta.resample[,2]),type='l',xlab=expression(beta[1]),ylab='approximate posterior density')
```


density(x = beta.resample[, 1])
approximate posterior density
0
2
4
6
8
10
3.30
3.40
3.50
β0
density(x = beta.resample[, 2])
approximate posterior density
0
1
2
3
4
5
6
7
−0.4
−0.3
−0.2
−0.1
0.0
β1