Bayesian Learning

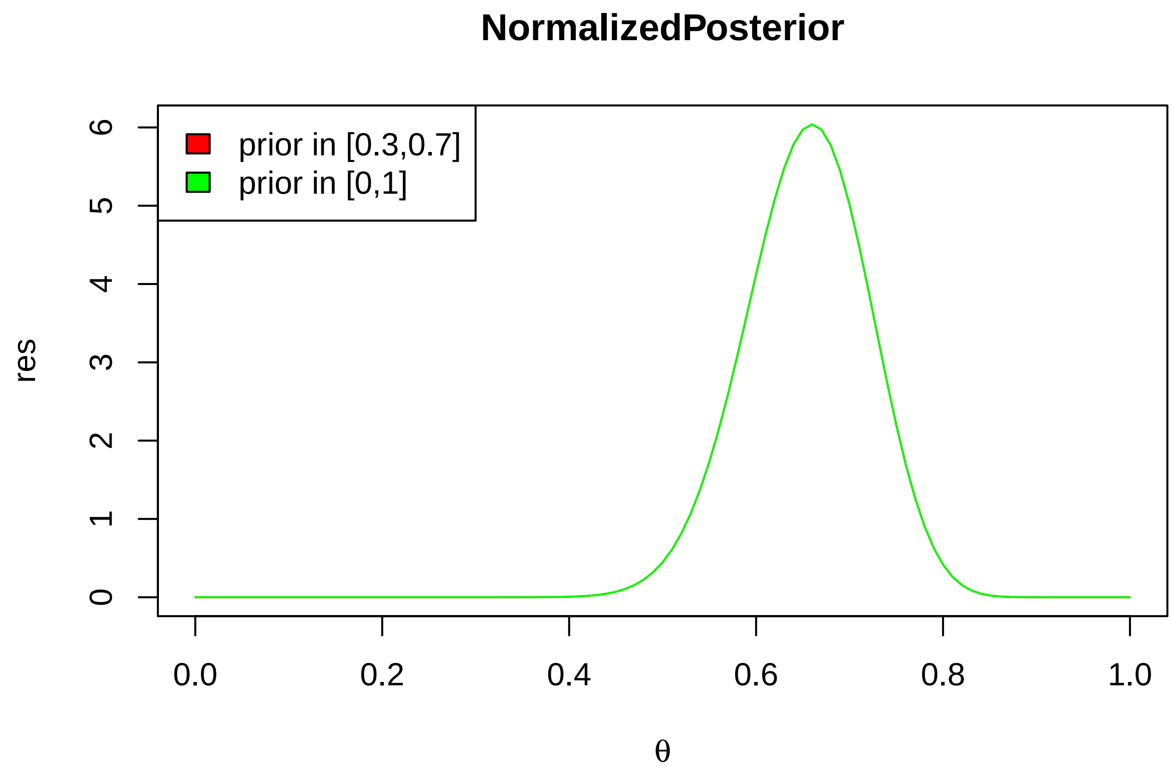
Mohsen Pirmoradiyan

6/4/2020

1

###b

```
posterior = function(theta, x){
  return(dbeta(theta, x+1, 51-x))

}

thetaGrid = seq(0,1,0.01)

# prior in [0.3, 0.7] p(theta) = 1/(0.7 - 0.3)

res =  sapply(1:length(thetaGrid), function(i)posterior(thetaGrid[i], x=33))

plot(thetaGrid, res, type = "l", col="red", xlim =c(0,1), main = "NormalizedPosterior",
     xlab = expression(theta))

# prior in [0, 1] p(theta) = 1
res2 = sapply(1:length(thetaGrid), function(i)posterior(thetaGrid[i], x=33))

lines(thetaGrid, res2/(0.01*sum(res2)), col="green" )

legend("topleft",
       legend = c("prior in [0.3,0.7]", "prior in [0,1]"),
       fill = c("red", "green"))
```



c

```r
print("\nPrior in [0.3, 0.7]: ")
```

```
## [1] "\nPrior in [0.3, 0.7]: "
```

```r
cat("Pr(posterior < 0.5: ", sum(res < 0.5)/length(res), "\n\n\n")
```

```
## Pr(posterior < 0.5:  0.7128713
```

```r
print("Prior in [0, 1]: ")
```

```
## [1] "Prior in [0, 1]: "
```

```r
cat("Pr(posterior < 0.5: ", sum(res2 < 0.5)/length(res2), "\n")
```

```
## Pr(posterior < 0.5:  0.7128713
```

When the prior is uniform the normalized posterior for that will be the same regardless of the prior

2

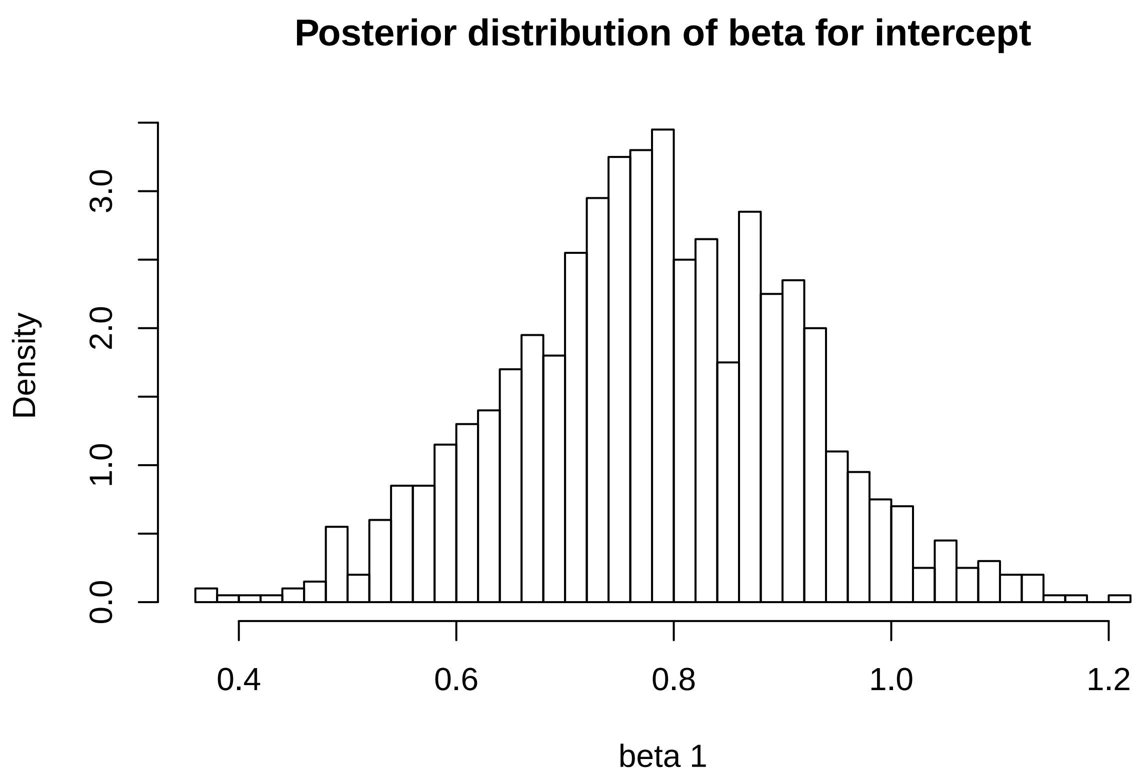
```r
load(file = 'titanic.RData')


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

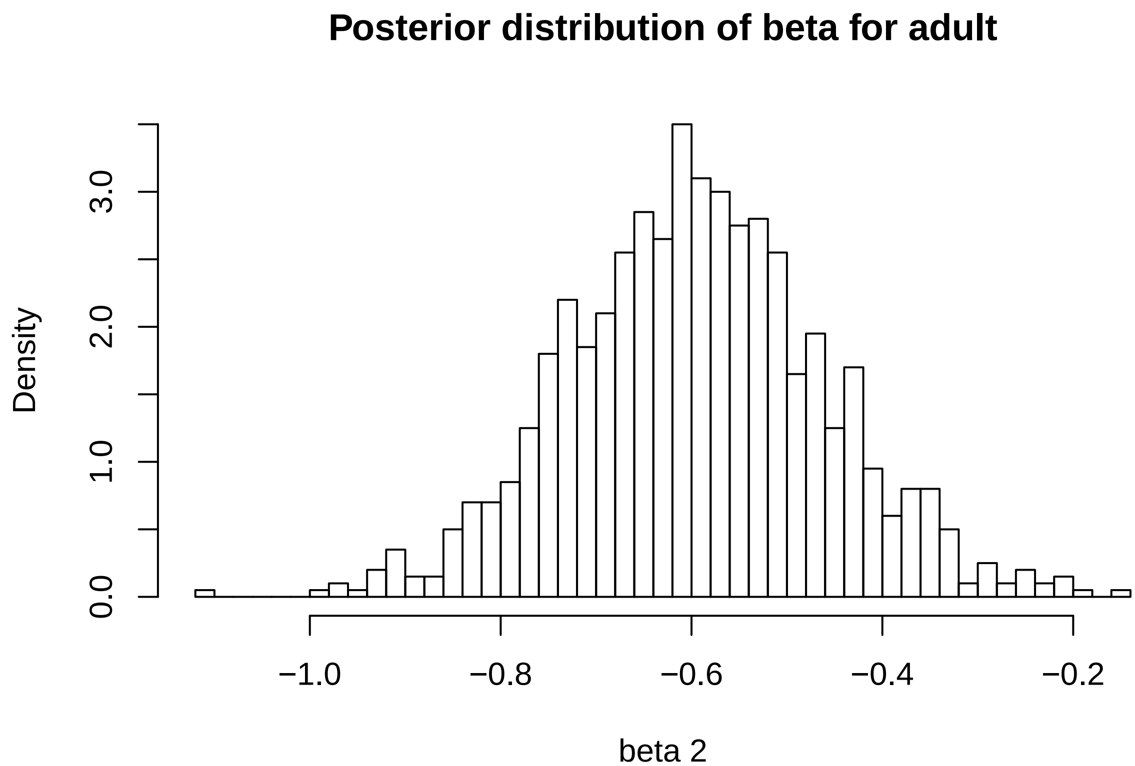
```r
  #
  # beta | tau ~ N(mu_0, tau^2*I)
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones if you want an intercept.
  #   mu_0 - prior mean for beta
  #   tau - prior standard deviation for beta
  #   nIter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

  # Prior
  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)
}
```

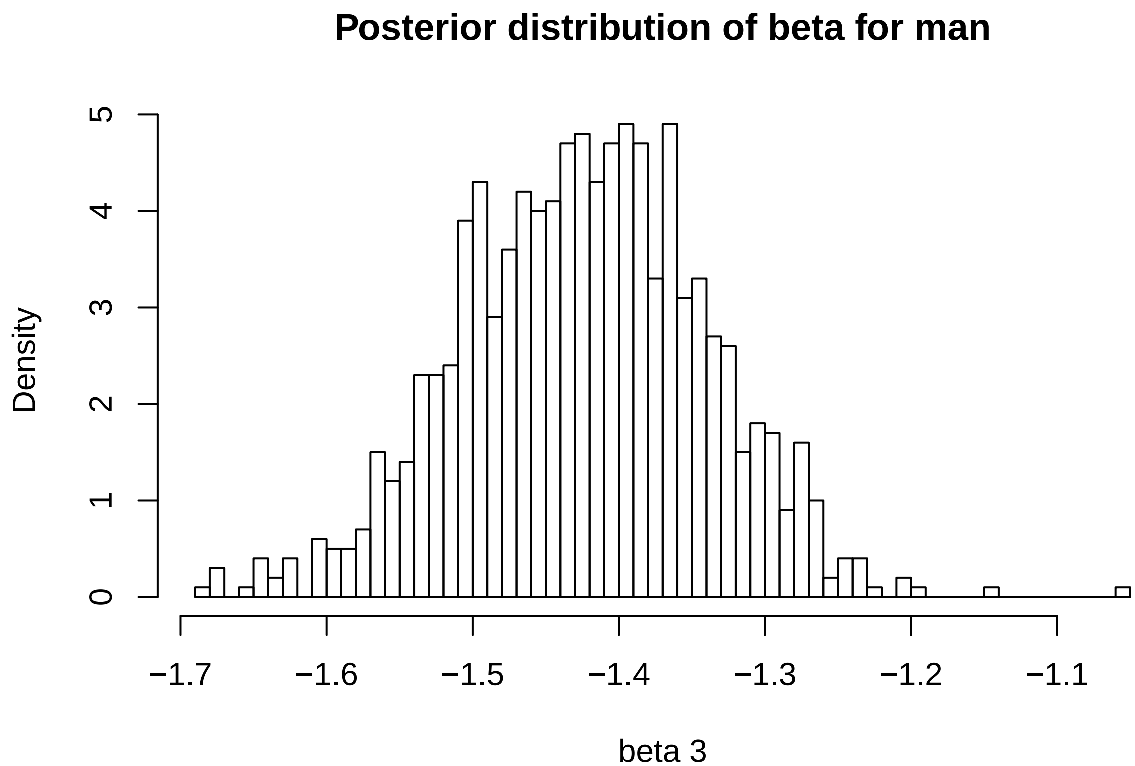
a

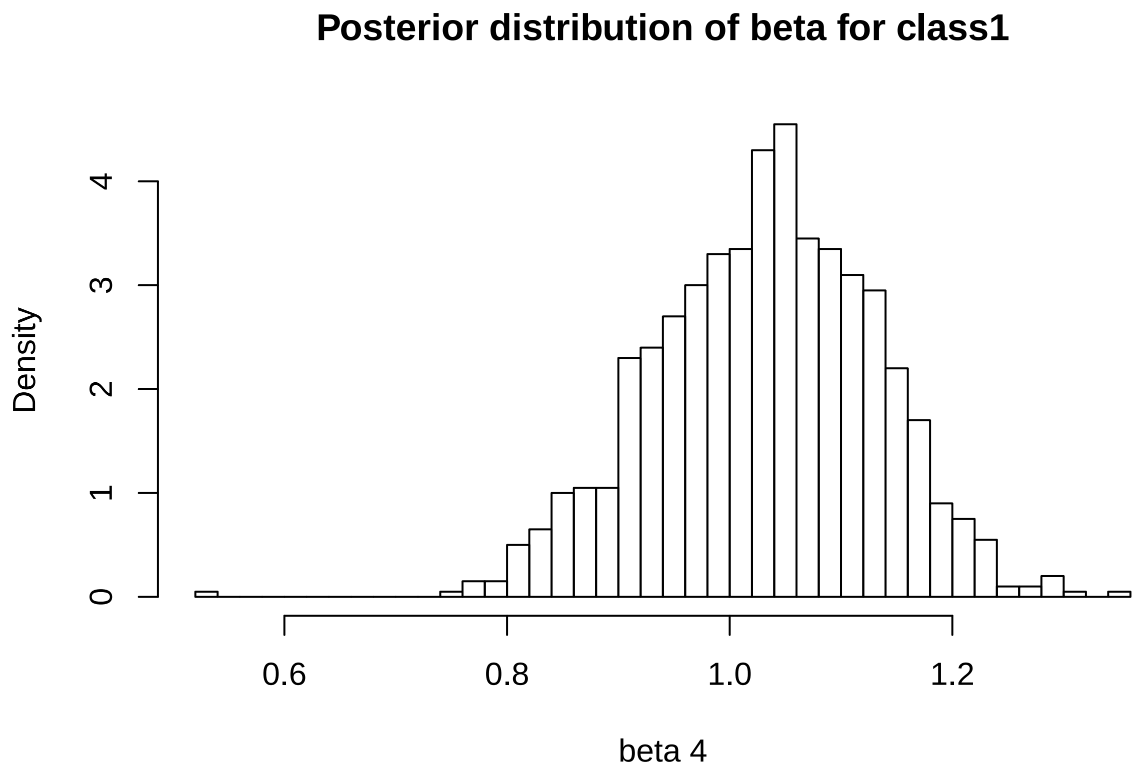
```r
y = as.vector(titanic$survived)
x = as.matrix(titanic[,-1])
```

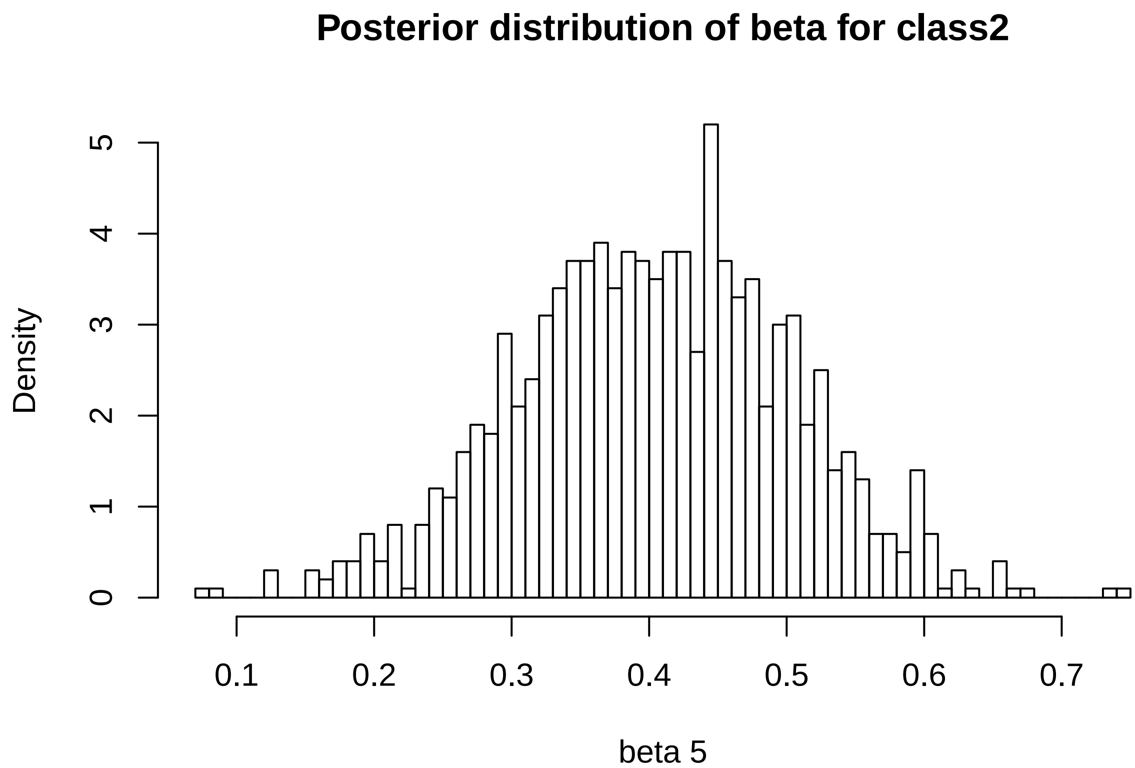
```r
covNames = colnames(x)
nPara = 5
beta = BayesProbReg(y=y, X=x,
                       mu_0=rep(0,5),
                       tau = 50,
                       nIter = 1000)


#Marginal Distribution
for(i in 1:nPara){
  hist(beta[,i], 50, freq=FALSE,
       main = paste("Posterior distribution of beta for", covNames[i]),
       xlab = paste(expression(beta), i))
}
```




Posterior distribution of beta for adult
Density
0.0
1.0
2.0
3.0
−1.0
−0.8
−0.6
−0.4
−0.2
beta 2


Posterior distribution of beta for man
Density
0
1
2
3
4
5
−1.7
−1.6
−1.5
−1.4
−1.3
−1.2
−1.1
beta 3


Posterior distribution of beta for class1
Density
0
1
2
3
4
0.6
0.8
1.0
1.2
beta 4



###b

Point estimate for a linear loss function would be the Median

```
for (i in 1:5) {
  cat("Point estimate for beta of ", covNames[i], ": ",
      median(beta[,i]),"\n\n")

}
```

```
## Point estimate for beta of  intercept :  0.7806523
##
## Point estimate for beta of  adult :  -0.5968705
##
## Point estimate for beta of  man :  -1.422684
##
## Point estimate for beta of  class1 :  1.036072
##
## Point estimate for beta of  class2 :  0.4056117
```

c

```
pr = numeric(nrow(beta))
for (i in 1:nrow(beta)) {
  if(beta[i,2]+beta[i,5] > 0){
    pr[i] = 1
  }else{
```

```r
    pr[i] = 0
  }
}


cat("Pr((Beta2 + Beta5) > 0) = ", sum(pr)/length(pr))
```

```
## Pr((Beta2 + Beta5) > 0) =  0.116
```

This probabilit says that if a person is adult and this person belong to the class1, it is roughly 11% probable to be survived

3

a

```r
y = c(5, 3, 17, 8)
x = c(log(20), log(20), log(50), log(40))

logPost = function(beta){
  logLik = sum(dpois(y, exp(x*beta), log = TRUE))
  logPrior = dnorm(beta, 1, 0.1, log = TRUE)
  return(logLik + logPrior)
}

initVal = 1
OptimResults<-optim(initVal,logPost,
                    method=c("BFGS"),
                    control=list(fnscale=-1),hessian=TRUE)
betaMean=OptimResults$par

Var = -solve(OptimResults$hessian)

cat("Posterior Mean :", betaMean, "\n\n")
```

```
## Posterior Mean : 0.6875749
```

```r
cat("standard deviation: ", sqrt(Var))
```

```
## standard deviation:  0.03958642
```

b

```r
loss = function(x,y){
  4 + (exp(x)/50) - sqrt(y)

}

cost = c(20, 40)

x = log(cost)

Loss = matrix(0, 10000, length(x))
for (i in 1:10000) {
  for (j in 1:length(x)) {
```

```r
    beta = rnorm(1,betaMean, sqrt(Var))
    pred = rpois(1,exp(beta*x[j]))
    Loss[i,j] = loss(x[j], pred)


}


}

cat("Expected value for loss when cost = 20: ", mean(Loss[,1]), "\n")
```

```
## Expected value for loss when cost = 20:  1.644435
```

```r
cat("Expected value for loss when cost = 40: ", mean(Loss[,2]), "\n")
```

```
## Expected value for loss when cost = 40:  1.26471
```

According to the expected values, we select the one with the lowest loss, so the decision would be keeping cost at 40 milion

4

```r
y1 = 0.75 * dnorm(10, 14, sqrt(4 + 4/20)) * dnorm(250, 300, sqrt(2500 + 2500/20))
y0 = 0.25 * dnorm(10, 12, sqrt(4 + 4/20)) * dnorm(250, 280, sqrt(2500 + 2500/20))

unNorm = c(y1, y0)
print("UnNormalized Prob:")
```

```
## [1] "UnNormalized Prob:"
```

```r
print(unNorm)
```

```
## [1] 0.0001051133 0.0001982958
```

```r
print("************")
```

```
## [1] "************"
```

```r
Normalized = unNorm/sum(unNorm)
names(Normalized) = c("Feamale", "Male")
print("Normalized:")
```

```
## [1] "Normalized:"
```

```r
print(Normalized)
```

```
##   Feamale      Male
## 0.3464408 0.6535592
```

Given the new measurements for length and weight and the previous data, the probability that the new fish is female is 0.35