Assignment2.R

snallfot

2023-05-09

# Metropolis Random Walk for Poisson regression.   
#Consider the following Poisson regression model y\_i|beta(iid) ~ Poisson[exp(x\_i\_transpose%\*%β)], i=1,..,n  
# where y\_i is the count for the ith observation in the sample   
# and xi is the p-dimensionalvector with covariate observations for the ith observation.   
# Use the data set eBayNumberOfBidderData.dat. This dataset contains observations from 1000 eBay auctions of coins.   
# The response variable is nBids and records the number of bids in each auction.   
# The remaining variables are features/covariates (x):  
# Const (for the intercept)  
# PowerSeller (equal to 1 if the seller is selling large volumes on eBay)  
# VerifyID (equal to 1 if the seller is a verified seller by eBay)  
# Sealed (equal to 1 if the coin was sold in an unopened envelope)  
# MinBlem (equal to 1 if the coin has a minor defect)  
# MajBlem (equal to 1 if the coin has a major defect)  
# LargNeg (equal to 1 if the seller received a lot of negative feedback from customers)  
# LogBook (logarithm of the book value of the auctioned coin according to expert sellers. Standardized)  
# MinBidShare (ratio of the minimum selling price (starting price) to the book value. Standardized).  
  
# (a) Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data   
# [Hint: glm.R, don't forget that glm() adds its own intercept so don't input the covariate Const].   
# Which covariates are significant?  
  
library(mvtnorm)  
  
ebay\_data <- (read.table("eBayNumberOfBidderData.dat", header=TRUE))  
Xnames <- names(ebay\_data[,2:ncol(ebay\_data)]) #Names of columns  
n\_obs <- dim(ebay\_data)[1] #number of observations  
data <- ebay\_data[,-2] #training and test data without Const  
#y <- as.numeric(data[,1]) #target output  
  
# maximum likelihood estimator of beta  
model = glm(formula=nBids~., data=data, family=poisson())  
summary(model)

##   
## Call:  
## glm(formula = nBids ~ ., family = poisson(), data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.5800 -0.7222 -0.0441 0.5269 2.4605   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.07244 0.03077 34.848 < 2e-16 \*\*\*  
## PowerSeller -0.02054 0.03678 -0.558 0.5765   
## VerifyID -0.39452 0.09243 -4.268 1.97e-05 \*\*\*  
## Sealed 0.44384 0.05056 8.778 < 2e-16 \*\*\*  
## Minblem -0.05220 0.06020 -0.867 0.3859   
## MajBlem -0.22087 0.09144 -2.416 0.0157 \*   
## LargNeg 0.07067 0.05633 1.255 0.2096   
## LogBook -0.12068 0.02896 -4.166 3.09e-05 \*\*\*  
## MinBidShare -1.89410 0.07124 -26.588 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 2151.28 on 999 degrees of freedom  
## Residual deviance: 867.47 on 991 degrees of freedom  
## AIC: 3610.3  
##   
## Number of Fisher Scoring iterations: 5

#the z-value shows that VerifyID, Sealed, MajBlem, LogBook and MinBidShare are significant  
  
# (b) Let's do a Bayesian analysis of the Poisson regression. Let the prior be beta ∼ N[0, 100\*(X\_T%\*%X)]  
# where X is the n × p covariate matrix. This is a commonly used prior, which is called Zellner's g-prior.   
# Assume first that the posterior density is approximately multivariate normal:  
# beta|y ~ N(beta\_tilde, Jy^-1(beta\_tilde))  
#where β\_tilde is the posterior mode and Jy(beta\_tilde) is the negative Hessian at the posterior mode.   
#beta\_tilde and Jy(beta\_tilde) can be obtained by numerical optimization (optim.R)  
# exactly like you already did for the logistic regression in Lab 2   
# (but with the log posterior function replaced by the corresponding one for the Poisson model,   
#which you have to code up.).  
  
#function for Poisson logPosterior  
logPostPoisson = function(beta,y,X,mu,Sigma) {  
 beta <- matrix(beta, nrow=1)  
 linPred <- X%\*%t(beta)  
 logLik <- sum( - exp(linPred) + linPred\*y - log(factorial(y)) ) #loglikelihood  
 logPrior <- dmvnorm(beta, mu, Sigma, log=TRUE); #prior  
 return(logLik + logPrior) #prior and likelihood are added since we use log  
}  
  
#initialize variables  
y <- as.numeric(ebay\_data[,1]) #target output  
X <- as.matrix(ebay\_data[,-1]) #training data  
X <- apply(X, c(1, 2), as.numeric) #convert from characters to doubles  
n\_col <- dim(X)[2] #number of columns in training data  
initVal <- rep(0,n\_col) #initial values for Beta  
mu <- as.matrix(rep(0,n\_col)) # Prior mean vector for Beta  
Sigma <- 100\*solve(t(X)%\*%X) # Prior covariance matrix for Beta  
logPost <- logPostPoisson; #function to be minimized  
  
  
set.seed(123)  
OptimRes <- optim(initVal,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)  
  
# Printing the results of mean and variance of normally distributed Beta   
OptimBeta <- OptimRes$par  
names(OptimBeta) <- Xnames # Naming the coefficient by covariates  
PostCov <- solve(-OptimRes$hessian) #get covariance matrix by inverting the negative hessian  
approxPostStd <- sqrt(diag(PostCov)) # Computing approximate standard deviations.  
names(approxPostStd) <- Xnames # Naming the coefficient by covariates  
print('Posterior beta values:')

## [1] "Posterior beta values:"

print(OptimBeta)

## Const PowerSeller VerifyID Sealed Minblem MajBlem   
## 1.06984118 -0.02051246 -0.39300599 0.44355549 -0.05246627 -0.22123840   
## LargNeg LogBook MinBidShare   
## 0.07069683 -0.12021767 -1.89198501

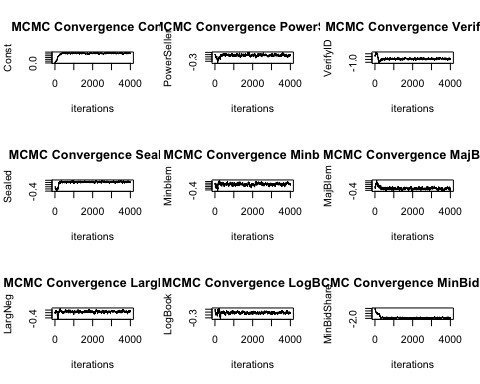
print('Posterior standard deviation:')

## [1] "Posterior standard deviation:"

print(approxPostStd)

## Const PowerSeller VerifyID Sealed Minblem MajBlem   
## 0.03074837 0.03678418 0.09227871 0.05057448 0.06020470 0.09146070   
## LargNeg LogBook MinBidShare   
## 0.05634767 0.02895635 0.07109682

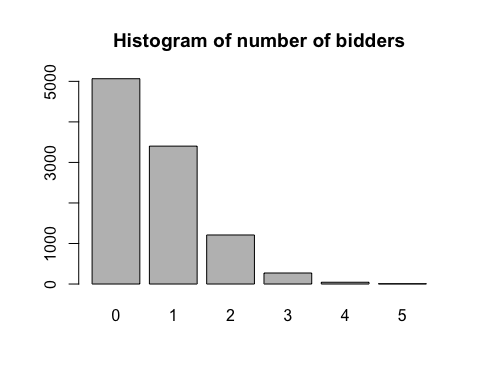
#the result seems reasonable since they are similar to a)  
  
# (c) Let's simulate from the actual posterior of beta using the Metropolis algorithm  
# and compare the results with the approximate results in b).   
# Program a general function that uses the Metropolis algorithm to generate random draws from an  
# arbitrary posterior density. In order to show that it is a general function for  
# any model, we denote the vector of model parameters by θ. Let the proposal density   
# be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):  
# θp|θ(i−1) ∼ N(θ(i−1), c · Σ) where Σ = Jy^−1(beta?tilde) was obtained in b).   
# The value c is a tuning parameter and should be an input to your Metropolis function.   
# The user of your Metropolis function should be able to supply her own posterior density function, not  
# necessarily for the Poisson regression, and still be able to use your Metropolis  
# function. This is not so straightforward, unless you have come across function  
# objects in R. The note HowToCodeRWM.pdf in Lisam describes how you can do this in R.  
  
#Now, use your new Metropolis function to sample from the posterior of β  
# in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.  
  
RWMSampler <- function(oldTheta, logPostFunc, c, PostCov, ... ) {  
 proposedTheta <- rmvnorm(1, mean=oldTheta, sigma=c\*PostCov)  
 alpha <- min(1, exp(logPostFunc(proposedTheta, ...) - logPostFunc(oldTheta, ...) ))  
 t <- runif(1)  
 if (alpha>t) {  
 return(list(proposedTheta, alpha))  
 } else {  
 return(list(oldTheta, alpha))  
 }  
}  
  
nDraws <- 4000  
c <- 0.7  
beta <- matrix(0, nDraws, n\_col) #matrix to fill with samples from RWM  
alpha\_vector <- rep(0, nDraws) #will be used to calculate mean alpha  
logPost <- logPostPoisson  
  
  
for(i in 1:nDraws) {  
 output <- RWMSampler(beta[i,], logPost, c, PostCov, y, X, mu, Sigma)  
 alpha\_vector[i] <- output[[2]]  
 if(i<nDraws) {  
 beta[i+1,] <- output[[1]]  
 }  
}  
  
iterations=seq(1,nDraws,1)  
par(mfrow=c(3,3))  
for (i in 1:n\_col) {  
 plot(iterations, beta[,i], type="l", main=paste("MCMC Convergence", Xnames[i]),  
 ylab=Xnames[i])  
}



par(mfrow=c(1,1), new=FALSE)  
  
# Calculate average alpha  
average\_alpha <- mean(alpha\_vector)  
average\_alpha #0.2531, has average acceptance probability between 25-30%

## [1] 0.2621183

# (d) Use the MCMC draws from c) to simulate from the predictive distribution of  
# the number of bidders in a new auction with the characteristics below.   
#Plot the predictive distribution. What is the probability of no bidders in this new auction?  
# PowerSeller = 1  
# VerifyID = 0  
# Sealed = 1  
# MinBlem = 0  
# MajBlem = 1  
# LargNeg = 0  
# LogBook = 1.2  
# MinBidShare = 0.8  
  
nDraws <- 10000  
x <- c(1,1,0,1,0,1,0,1.2,0.8)  
post\_beta <- beta[(1000:nrow(beta)),] #select samples after convergence  
mean <- exp(post\_beta%\*%x) #vector of means  
samples <- rpois(nDraws, mean)  
barplot(table(samples), main="Histogram of number of bidders")



prob\_no\_bidders <- sum(samples==0)/sum(samples) #probability of no bidders  
print(prob\_no\_bidders)

## [1] 0.739524

#the probability of no bidders is 74%