Q2 report

Jonathan Dorairaj,Yi Hung Chen

2023-05-15

Metropolis Random Walk for Poisson regression

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 signifcant?

```
ebay_data <- read.table("eBayNumberOfBidderData.dat", header = TRUE)

model1 <- glm(formula = nBids ~ .,data = ebay_data[,-2],family = 'poisson')

summary(model1)
```

```
##
## Call:
## glm(formula = nBids ~ ., family = "poisson", data = ebay_data[,
##     -2])
##
## 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 estimates of beta are 1.0724421, -0.0205408, -0.3945165, 0.4438426, -0.0521983, -0.2208712, 0.0706725, -0.1206776, -1.8940966

The significant covariates are **'VerfiyID'**,**'Sealed'**,**'LogBook'** and **'MinBidShare'** and **'Intercept'**.**'MajBlem'** is also significant but to a less degree compared to the ones mentioned before.

b) Let's do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim \mathcal{N}[0, 100 \cdot (X^T X)^{-1}]$, where X is the n $\times$ 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:

$$\beta \mid y \sim \mathcal{N}\left(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})\right)$$

where $\tilde{\beta}$ is the posterior mode and $J_{\mathbf{y}}(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_{\mathbf{y}}(\tilde{\beta})$ 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.).

The Poisson distribution density is given by :

$$p(y \mid x; \beta) = \frac{e^{y\beta' x} e^{-e^{\beta'}}}{y!}$$

The log likelihood of the poisson distribution is :

$$\ell(\beta \mid X, Y) = \log L(\beta \mid X, Y) = \sum_{i=1}^{n} \left(y\beta' x - e^{\beta' x} - \log y! \right)$$

```r
# Initialize values

n_cols <- ncol(ebay_data[,-1])
#remove 1st column since that is target variable and convert to matrix
# matrix of features
covariates <- as.matrix(ebay_data[,-1])
labels <- as.matrix(ebay_data[,1])
mu <- rep(0, n_cols)
initVal <- matrix(0, n_cols, 1)
Sigma <- as.matrix(100 * solve(t(covariates)%*%covariates))

LogPosteriorFunc <- function(betas, X, y, mu, Sigma){
  log_prior <- dmvnorm(betas, mu, Sigma, log=TRUE)
  log_likelihood <- sum(X%*%betas * y - exp(X%*%betas) -log(factorial(y)))
  res <- log_prior + log_likelihood
  return(res)
}

# Optimizer
OptimRes <- optim(initVal, LogPosteriorFunc, gr = NULL, y = labels, X = covariates,
                  mu = mu, Sigma = Sigma, method=c("BFGS"),
                  control=list(fnscale=-1), hessian=TRUE)
```

```r
beta_mode <- OptimRes$par
jacobian <- OptimRes$hessian
inv_jacobian <- -solve(jacobian)

beta_draws <- as.matrix(rmvnorm(10000,mean = beta_mode,sigma = inv_jacobian))
beta_estimate <- colMeans(beta_draws)
```

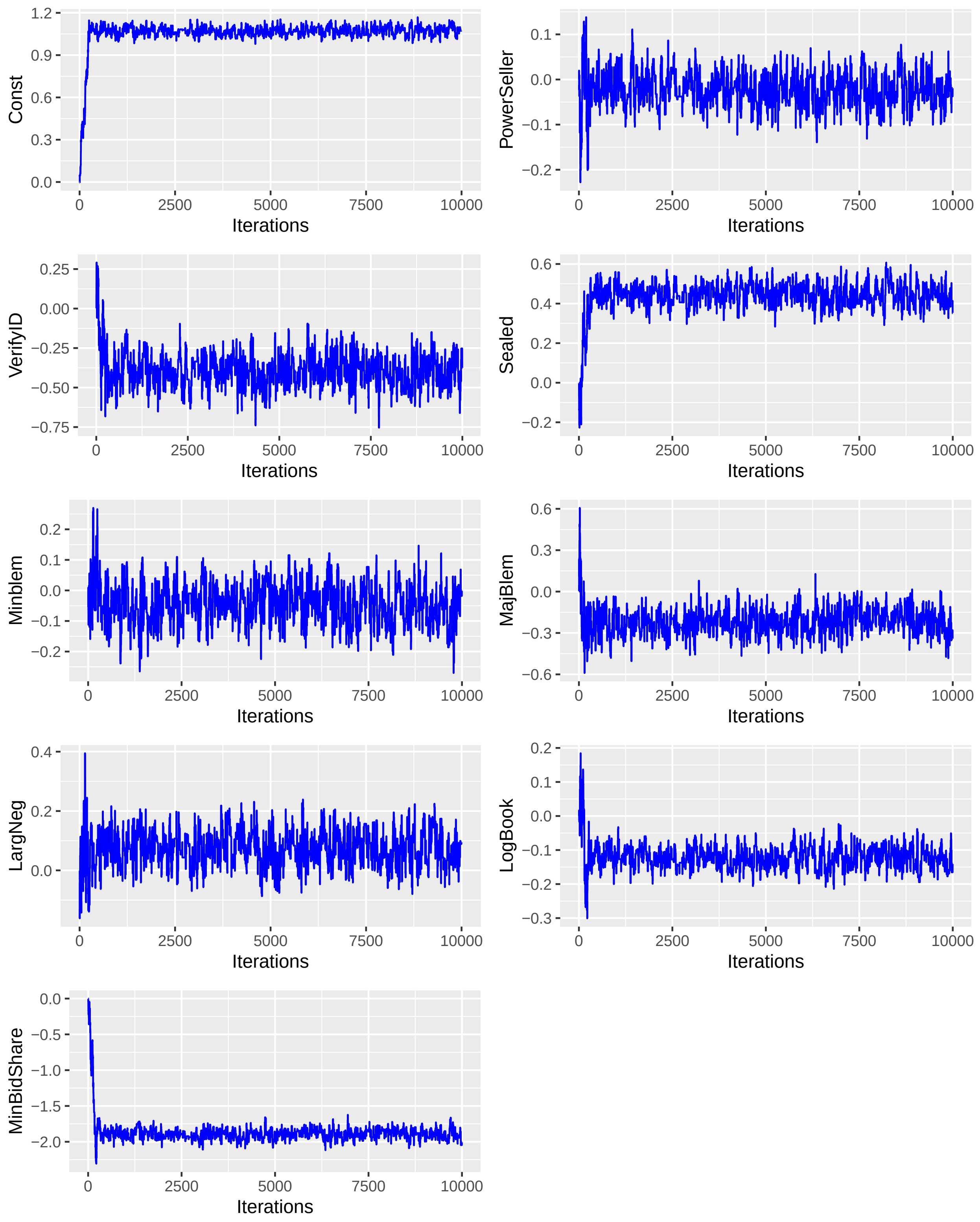
```r
hist(beta_draws,breaks = 50,main = 'Histogram of Posterior Draws',xlab = 'Betas')
```

c) Let's simulate from the actual posterior of β 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):

$$\theta_p \mid \theta^{(i-1)} \sim N\left(\theta^{(i-1)}, c \cdot \Sigma\right)$$

where $\Sigma = J_{\mathbf{y}}^{-1}(\tilde{\beta})$ 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 HowToCo deRWM.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.

```r
MetHas_RandomWalk <- function(nDraws,fun,mu,Sigma,c){

  #initialize matrix
  draw_matrix <- matrix(0,nrow = nDraws,ncol = n_cols)
  #initialize first row to mu
  draw_matrix[1,] <- mu

  for(i in 2:nDraws){
    # sample from multivariate normal distribution
    proposed_sample <- as.vector(rmvnorm(n = 1,mean = draw_matrix[i-1,],
                                          sigma = c*as.matrix(Sigma)))
    #print(proposed_sample)
    # IMPORTANT : the log is inside the posterior function
    log_acceptance_prob <- exp(fun(proposed_sample)- fun(draw_matrix[i-1,]))

    #random sample
    u <- runif(1)
    # calculate acceptance probability
    a <- min(1,log_acceptance_prob)


    if(u <= a){
      #accept sample
      draw_matrix[i,] <- proposed_sample
    }
    else{
      # stay at same values from previous draw
      draw_matrix[i,] <- draw_matrix[i-1,]
    }

  }
  return(draw_matrix)
}

# this function we pass to MetHas Algorithm, can be changed to another posterior density
logPostFunc <- function(theta){
  res <- dmvnorm(theta,mean = beta_estimate,sigma = inv_jacobian,log = TRUE)
  if(is.na(res)){
    print(theta)
  }
  return(res)
}

df <- MetHas_RandomWalk(nDraws = 10000,fun = logPostFunc,mu = rep(0,n_cols),
                        Sigma = inv_jacobian,c = 1)
# assign colnames
colnames(df) <- colnames(ebay_data)[2:10]

## plotting
plot_list <- list()
for (col in colnames(df)) {
  # Plot iterations vs every column
  p <- ggplot(data = as.data.frame(df), aes_string(x = 1:nrow(df), y = col)) +
    geom_line(col = 'blue') +
    labs(x = "Iterations", y = col)

  #show plot
  #print(p)
  plot_list[[col]] <- p
}

# Arranging in 1 fig
grid.arrange(grobs = plot_list, ncol = 2)
```



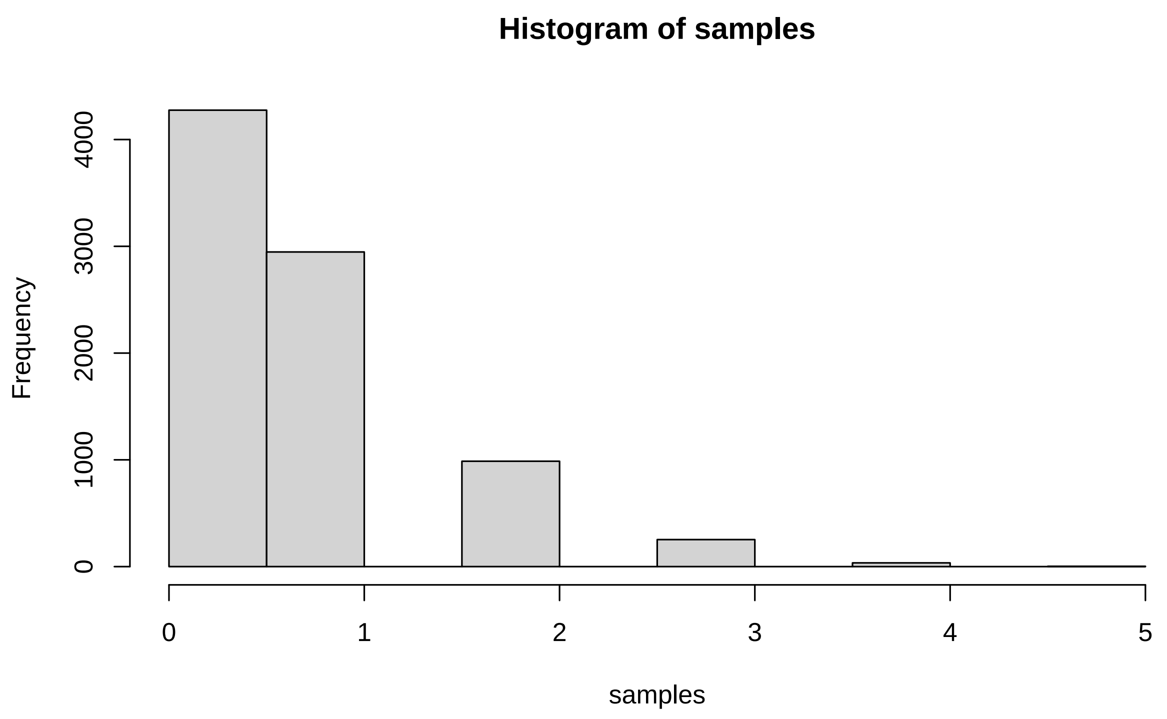
Assessing the plots of the 9 column variables vs Iterations, we see that convergence usually occurs after 1500 iterations. (burn-in period)

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?

- Power Seller = 1
- VerifyID = 0
- Sealed = 1
- MinBlem = 0
- MajBlem = 1
- LargNeg = 0
- **Log** Book = 1.2
- MinBidShare = 0.8

$$\text{Mean} = \lambda = e^{\beta' \cdot x}$$

```r
# extra 1 at the start for the intercept - Const
new_data <- c(1,1,0,1,0,1,0,1.2,0.8)

# lambda =  e^Beta*x
# discarding first 1500 samples as burn-in period
lambda <- exp(df[-c(1:1500),] %*% new_data)

samples <- c()

for (i in 1:nrow(df[-c(1:1500),])) {
  #sample from each row of df to get the predictive distribution based on the
  #posterior betas
  samples[i] <- rpois(1,lambda = lambda[i])

}

hist(samples)
```



```r
res <- length(samples[samples == 0])/length(samples)
```

The probability of have zero bidders in the new auction is 50.2941176%.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(fig.align = "center")
knitr::opts_chunk$set(fig.width=8, fig.height=5)
knitr::opts_chunk$set(warning=FALSE)
library(mvtnorm)
library(ggplot2)
library(gridExtra)



ebay_data <- read.table("eBayNumberOfBidderData.dat", header = TRUE)

model1 <- glm(formula = nBids ~ .,data = ebay_data[,-2],family = 'poisson')

summary(model1)
# Initialize values

n_cols <- ncol(ebay_data[,-1])
#remove 1st column since that is target variable and convert to matrix
# matrix of features
covariates <- as.matrix(ebay_data[,-1])
labels <- as.matrix(ebay_data[,1])
mu <- rep(0, n_cols)
initVal <- matrix(0, n_cols, 1)
Sigma <- as.matrix(100 * solve(t(covariates)%*%covariates))

LogPosteriorFunc <- function(betas, X, y, mu, Sigma){
  log_prior <- dmvnorm(betas, mu, Sigma, log=TRUE)
  log_likelihood <- sum(X%*%betas * y - exp(X%*%betas) -log(factorial(y)))
  res <- log_prior + log_likelihood
  return(res)
}

# Optimizer
OptimRes <- optim(initVal, LogPosteriorFunc, gr = NULL, y = labels, X = covariates,
                  mu = mu, Sigma = Sigma, method=c("BFGS"),
                  control=list(fnscale=-1), hessian=TRUE)

beta_mode <- OptimRes$par
jacobian <- OptimRes$hessian
inv_jacobian <- -solve(jacobian)

beta_draws <- as.matrix(rmvnorm(10000,mean = beta_mode,sigma = inv_jacobian))
beta_estimate <- colMeans(beta_draws)


hist(beta_draws,breaks = 50,main = 'Histogram of Posterior Draws',xlab = 'Betas')
MetHas_RandomWalk <- function(nDraws,fun,mu,Sigma,c){

  #initialize matrix
  draw_matrix <- matrix(0,nrow = nDraws,ncol = n_cols)
  #initialize first row to mu
  draw_matrix[1,] <- mu

  for(i in 2:nDraws){
    # sample from multivariate normal distribution
    proposed_sample <- as.vector(rmvnorm(n = 1,mean = draw_matrix[i-1,],
                                         sigma = c*as.matrix(Sigma)))
    #print(proposed_sample)
    # IMPORTANT : the log is inside the posterior function
    log_acceptance_prob <- exp(fun(proposed_sample)- fun(draw_matrix[i-1,]))

    #random sample
    u <- runif(1)
    # calculate acceptance probability
    a <- min(1,log_acceptance_prob)


    if(u <= a){
      #accept sample
      draw_matrix[i,] <- proposed_sample
    }
    else{
      # stay at same values from previous draw
      draw_matrix[i,] <- draw_matrix[i-1,]
    }

  }
  return(draw_matrix)
}

# this function we pass to MetHas Algorithm, can be changed to another posterior density
logPostFunc <- function(theta){
  res <- dmvnorm(theta,mean = beta_estimate,sigma = inv_jacobian,log = TRUE)
  if(is.na(res)){
    print(theta)
  }
  return(res)
}

df <- MetHas_RandomWalk(nDraws = 10000,fun = logPostFunc,mu = rep(0,n_cols),
                        Sigma = inv_jacobian,c = 1)
# assign colnames
colnames(df) <- colnames(ebay_data)[2:10]

## plotting
plot_list <- list()
for (col in colnames(df)) {
  # Plot iterations vs every column
  p <- ggplot(data = as.data.frame(df), aes_string(x = 1:nrow(df), y = col)) +
    geom_line(col = 'blue') +
    labs(x = "Iterations", y = col)

  #show plot
  #print(p)
  plot_list[[col]] <- p
}

# Arranging in 1 fig
grid.arrange(grobs = plot_list, ncol = 2)

# extra 1 at the start for the intercept - Const
new_data <- c(1,1,0,1,0,1,0,1.2,0.8)

# lambda =  e^Beta*x
# discarding first 1500 samples as burn-in period
lambda <- exp(df[-c(1:1500),] %*% new_data)

samples <- c()

for (i in 1:nrow(df[-c(1:1500),])) {
  #sample from each row of df to get the predictive distribution based on the
  #posterior betas
  samples[i] <- rpois(1,lambda = lambda[i])

}

hist(samples)

res <- length(samples[samples == 0])/length(samples)
```