732A91: Lab 4
Bayesian Learning

Sarah Alsaadi, Carles Sans Fuentes

May 24, 2017

Poisson regression-the MCMC way

Consider the following Poisson regression model

$$y_i|\beta \sim \text{Poisson}[\exp(\mathbf{x}_i^T \beta)], i = 1, \ldots, n,$$

where y_i is the count for the ith observation in the sample and $\mathbf{x}_i$ is the p-dimensional vector with covariate observations for the ith observation. The data set **eBayNumberOfBidderData.dat** 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 ($\mathbf{x}$):

- **Const** (for the intercept)
- **PowerSeller** (is the seller selling large volumes on eBay)
- **VerifyID**(is the seller verified by eBay?)
- **Sealed** (was the coin sold sealed in never opened envelope?)
- **MinBlem** (did the coin have a minor defect?)
- **MajBlem** (a major defect?)
- **LargNeg** (did the seller get a lot of negative feedback from customers?)
- **LogBook** (logarithm of the coins book value according to expert sellers. Standardized)
- **MinBidShare** (a variable that measures ratio of the minimum selling price (starting price) to the book value. Standardized).

(a) Using **glm** in R, we obtain the following results:

```
Call:
glm(formula = nBids ~ 0 + ., family = poisson, data = ebay)

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|)
Const        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: 6264.01  on 1000  degrees of freedom
```

```
Residual deviance:  867.47  on  991  degrees of freedom
AIC: 3610.3

Number of Fisher Scoring iterations: 5
```

The output shows that the significant MLE of the β:s are those for the covariates Const, VeryfyID, Sealed, MayBlem, LogBook and MinBidShare.

(b) In this exercise we did a Bayesian analysis of the Poisson regression. We let $\beta \sim N[0, 100(\mathbf{X}^T\mathbf{X})^{-1}]$ apriori, where $\mathbf{X}$ is the $n \times p$ covariate matrix. Next we assumed that the posterior density is approximately multivariate normal:

$$\beta|\mathbf{y}, \mathbf{X} \sim N(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta}))$$

where $\tilde{\beta}$ is the posterior mode and $J_{\mathbf{y}}(\tilde{\beta}) = -\frac{\partial^2 lnp(\beta|\mathbf{y})}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. To obtain $\tilde{\beta}$ and $J_{\mathbf{y}}^{-1}(\tilde{\beta})$ we used numerical optimization (**optim.R**). The results of the estimated β:s are given in the table below. We also made histograms, one for each variable, of 10000 draws of β, the marginal distribution looks normal, see Figure 1.

Variable	$\tilde{\beta}$
Const	1.06984118
PowerSeller	-0.02051246
VerifyID	-0.39300599
Sealed	0.44355549
MinBlem	-0.05246627
MajBlem	0.22123840
LargNeg	0.07069683
LogBook	-0.12021767
MinBidShare	-1.89198501

Table 1: The obtained β:s through maximization of the posterior distirbution, one for each variable.

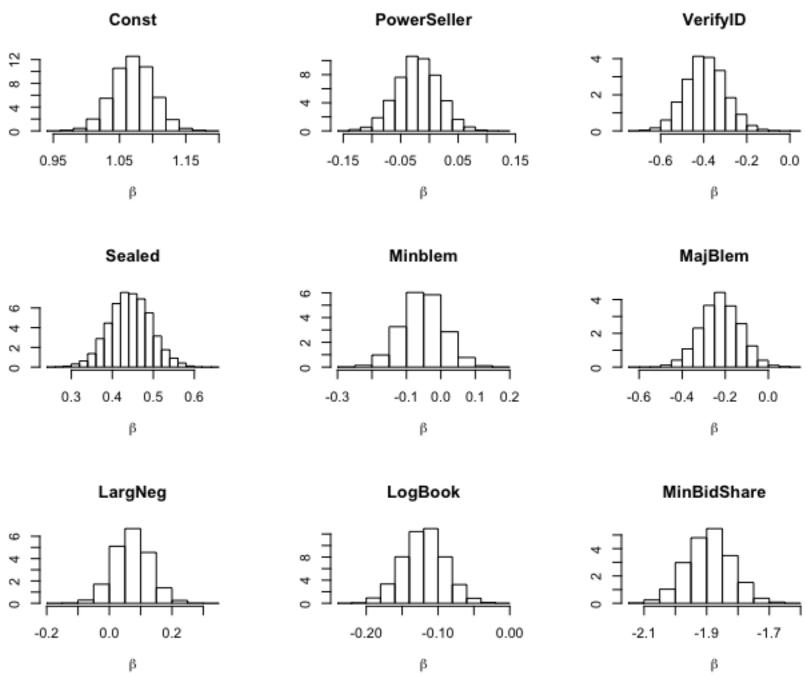

Figure 1: Histogram of the 10000 β:s drawn from the marginal posterior distribution

(c) In this exercise we simulate from the actual posterior of β using the Metropolis algorithm and compare with the approximate results in b). We program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. We use the multivariate normal density as proposal density:

$$\theta_p|\theta_c \sim N(\theta_c, \tilde{c} \cdot \Sigma)$$

where $\Sigma = J_{\mathbf{y}}^{-1}(\tilde{\beta})$ obtained in the previous exercise and θ_c is the current draw (hence the subscript c). The value $\tilde{c}$ is a tuning parameter and is an input to our Metropolis function (so that a user can change it). The user of our Metropolis function is able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use our Metropolis function.

First, one of the input arguments of our Metropolis function is called logPostFunc. logPostFunc is a function object that computes the log posterior density at any value of the parameter vector. This is needed when we compute the acceptance probability of the Metropolis algorithm. We program the log posterior density, since logs are more stable and avoids problems with too small or large numbers (overflow). Note that the ratio of posterior densities in the Metropolis acceptance probability can be written

$$\frac{p(\theta_p|\mathbf{y})}{p(\theta_c|\mathbf{y})} = \exp[log(p(\theta_p|\mathbf{y})) - log(p(\theta_c|\mathbf{y}))]$$

This is smart since the large or small common factors in $p(\theta_p|\mathbf{y})$ and $p(\theta_c|\mathbf{y})$ cancel out before we evaluate the exponential function (which can otherwise overflow).

Second, the first argument of our (log) posterior function is theta, the (vector) of parameters for which the posterior density is evaluated.

Third, the user's posterior density is also a function of the data and prior hyperparameters and those are supplied to the Metropolis function where we use the triple dot (...) argument which is like a wildcard for any parameters supplied by the user. This makes it possible to use the Metropolis function for any problem, even when a programmer don't know what the user's posterior density function looks like or what kind of data and hyperparameters being used in that particular problem.

Now, we use Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. We assess MCMC convergence by graphical methods. The parameters $\phi_j = \exp(\beta_j)$ are usually considered more interpretable than the β_j. We compute the posterior distribution of ϕ_j for all variables.

Figure 2 shows that the Metropolis algorithm seem to converge, with some burn in period for all the β:s.

In Table 2 we have the mean posterior of the simulated β:s which are very similar to the ones obtained through maximization.

Figure 3 shows the marginal posterior for each e^{β} obtained by the Metropolis algorit. The distribution looks normal.

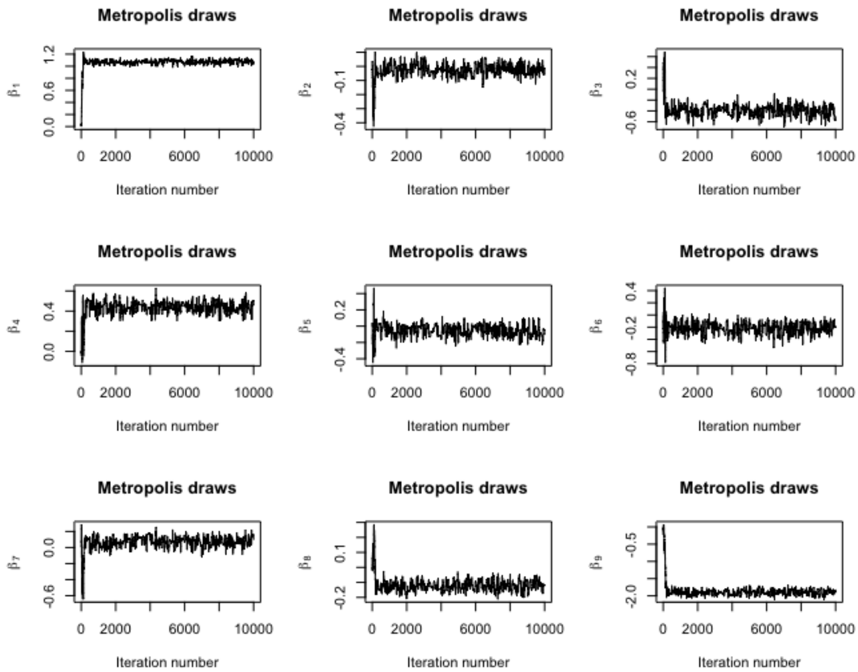

Figure 2: Plot of the 10000 β:s drawn from the marginal posterior distribution obtained by the Metropolis algorithm.

Variable	$\tilde{\beta}$
Const	1.06538266
PowerSeller	-0.02529706
VerifyID	-0.38552635
Sealed	0.43553156
MinBlem	-0.05410433
MajBlem	-0.22019352
LargNeg	0.06493763
LogBook	-0.12021880
MinBidShare	-1.88132053

Table 2: The posterior mean of the simulated β:s, simulated using the Metropolis algorithm, the values are almost identical as the ones obtained through maximization

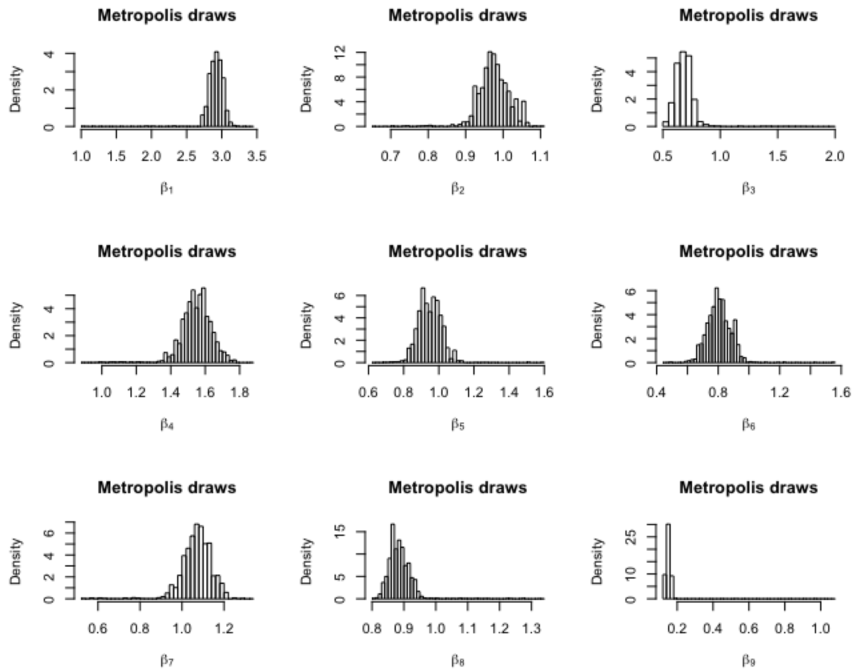

Figure 3: Histogram of the 10000 e^{β} drawn from the marginal posterior distribution obtained by the Metropolis algorithm.

(d) We 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. The histogram of the predictive distribution is shown in Figure 4. The probability of no bidders in this new auction is 0.3512.

- **PowerSeller**$= 1$
- **VerifyID**$= 1$
- **Sealed**$= 1$
- **MinBlem**$= 0$
- **MajBlem**$= 0$
- **LargNeg**$= 0$
- **LogBook**$= 1$
- **MinBidShare**$= 0.5$

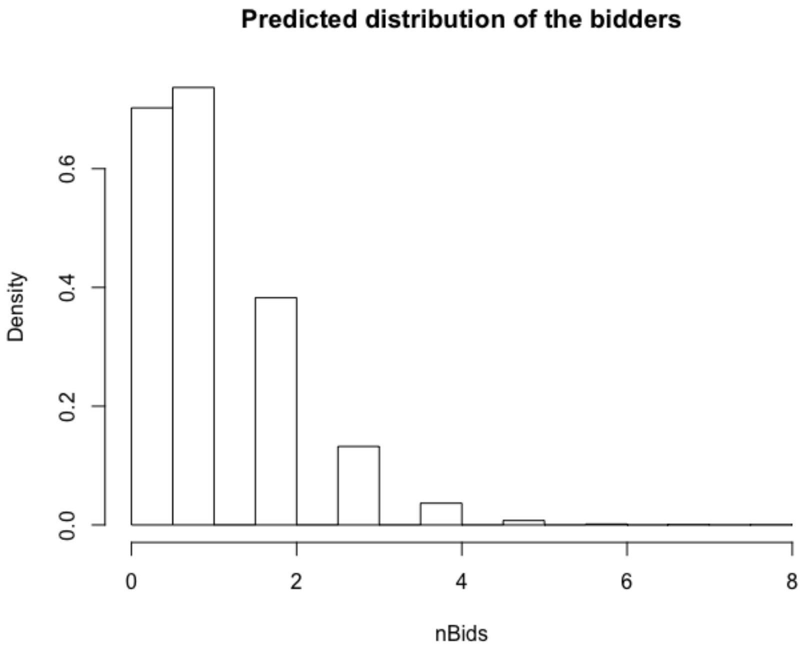

Figure 4: Histogram of 10000 simulated predicted values of the variable nBid with given characteristics

Contributions

All results and comments presented have been developed and discussed together by the members of the group.

Appendix

Poisson regression-the MCMC way

```

### LAB 4

#1
Data<- read.csv("C:/Users/Carles/Desktop/Bayesian learning/Part4/eBayNumberOfBidderData.dat.txt", sep = "")

##a
#### Variables

PoisModel<-glm(nBids~.-Const, family="poisson", data = Data)
logLik(PoisModel)


mysum<-summary(PoisModel)[["coefficients"]]



##Significants at a 95% level
significant <- mysum[which(mysum[,4]<0.05),]
non_significant <- mysum[which(mysum[,4]>=0.05),]
list(significant = significant, nonsignificant = non_significant)


##b
chooseCov <- 1:(length(names(Data))-1);  # Here we choose which covariates to include in the model


y <- as.vector(Data[,1]); # Data from the read.table function is a data frame. Let's convert y and X to vector and matrix.
X <- as.matrix(Data[,2:length(names(Data))]);
covNames <- names(Data)[2:length(names(Data))];
X <- X[,chooseCov]; # Here we pick out the chosen covariates.
covNames <- covNames[chooseCov];
nPara <- dim(X)[2]


##prior Beta
library(geoR)
library(mvtnorm)


mu_0    <- matrix(0, nPara,1)
Sigma_0 <- 100*solve(crossprod(X,X))
Beta_0  <- rmvnorm(1, mean = mu_0, sigma = Sigma_0)
InitVal <- matrix(0, ncol=nPara, nrow =1)


LogPostPois <- function(betaVect,y = y,X = X, mu =mu_0,Sigma= Sigma_0){
  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # The following is a more numerically stable evaluation of the log-likelihood in my slides:
  # logLik <- sum(y*log(pnorm(linPred)) + (1-y)*log(1-pnorm(linPred)) )
  logLik <- sum(linPred*y-exp(linPred))

  # evaluating the prior

  logPrior <- dmvnorm(betaVect, mu, Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)

}

OptimResults<-optim(InitVal,LogPostPois,gr=NULL,y = y,X = X, mu =mu_0,Sigma= Sigma_0 ,method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)

BetaCoef<- OptimResults$par
colnames(BetaCoef)<- covNames
J<--solve(OptimResults$hessian)

mycoefvar<- data.frame(Coefficients= as.vector(BetaCoef), variance = diag(J))
rownames(mycoefvar)<- covNames
mycoefvar
######C

set.seed(12345)

LogPostPoisson <- function(theta, priormu, priorsigma, X, Y, ...) {
```

```r
  require(mvtnorm)
  likelihood <- dpois(Y, lambda = as.vector(exp((X) %*% t(theta))), log = TRUE)
  prior <- dmvnorm(theta, mean = priormu, sigma = priorsigma, log=TRUE)
  return(sum(likelihood) + prior)
}


metropl<-function(logPostFunc, theta_0, constant, sigma, nIter,...){
  #initialize chain
  require(mvtnorm)
  theta<- matrix(NA, nrow = nIter+1, ncol = dim(sigma)[1])
  theta[1,]<- theta_0
  rej_rate<-0
  for(i in 2:(nIter+1)){
    new_theta<- rmvnorm(1, mean = theta[i-1,], sigma = constant*sigma)
    cur_theta<-t(as.matrix(theta[i-1,]))
    U<-runif(1,0,1)
    num<-logPostFunc(new_theta,...)+dmvnorm(cur_theta, mean =new_theta, sigma = sigma, log =
        TRUE)
    den<-logPostFunc(cur_theta,...)+dmvnorm(new_theta, mean =cur_theta, sigma = sigma, log =
        TRUE)

    if(U<min(1,exp(num-den))){
      theta[i,] <-new_theta
    }else{
      theta[i,] <-cur_theta
      rej_rate  <-rej_rate+1
    }}


  myres<- list(theta= theta, rej_rate= rej_rate/nIter)
  return(myres)
}


res<-metropl(logPostFunc=LogPostPoisson,
                    theta_0= rep(0,nPara),
                    constant= 0.6,
                    sigma = J ,
                    priormu= mu_0,
                    priorsigma= Sigma_0,
                    X= X, Y=y,
                    nIter= 10000)


res[[2]]

compbetas<- data.frame(OptimCoefficients= as.vector(BetaCoef), MCMCCoef =colMeans(res[[1]]))
rownames(compbetas)<- covNames
compbetas
##d
Xpred <- matrix(c(1, 1, 1, 1, 0, 0, 0, 1, 0.5), nrow = 1)
predsamples <- rpois(10000, lambda = exp(Xpred %*% t(res[[1]])))
hist(predsamples, freq = FALSE)

##probability of 0
length(which(predsamples==0))/length(predsamples)
```