# Assignment 12.b for **STATGR6103**

submitted to Professor Andrew Gelman

Advait Rajagopal

2 December 2016

## 1 Question 1

Computing problem: Consider probit regression, which is just like logistic except that the function $\text{logit}^{-1}$ is replaced by the normal cumulative distribution function. Set up and program variational Bayes for a probit regression with two coefficients (that is, $Pr(y_i = 1) = \Phi(a + bx_i)$, for $i = 1, ..., n$), using the latent-data formulation (so that $z_i \sim \text{N}(a + bx_i, 1)$ and $y_i = 1$ if $z_i > 0$ and 0 otherwise):

### 1.1 Part A

**Write the log posterior density (up to an arbitrary constant),** $p(a, b, z|y)$**.**
We know that the distribution of the latent variable $z_i$ is such that;

$$z_i \sim N(a + bx_i, 1)$$

$$y_i = \begin{cases} 1, z_i > 0 \\ 0, otherwise \end{cases}$$

The probability that $y_i = 1$ is given by the probit regression below;

$$Pr(y_i = 1) = \Phi(a + bx_i)$$

Where $\Phi$ is the cumulative distribution function of the normal distribution. Thus this is the likelihood function. And we know that the log posterior is proportional to the log likelihood, this means that the joint log posterior density of parameters $\alpha, \beta|y$ can be written as;

$$\log(p(\alpha, \beta|y)) \propto \sum_{i=1}^{n} \log(\Phi(a + bx_i))$$

Where we assign noninformative priors on $\alpha$ and $\beta$ such that ;

$$p(\alpha, \beta) \propto 1$$

### 1.2 Part B

**Assuming a variational approximation g that is independent in its** $n + 2$ **dimensions, determine the functional form of each of the factors in** $g$**.**

The variational approximation $g(\theta)$ is constructed iteratively with a goal to minimize Kullback-Leibler (KL) divergence from the target posterior distribution $p(\alpha, \beta|y)$. I explain what KL divergence is in the next section. For now it is important to note that the goal is to constrain the components of the parameters of $g$ to be independent. I use a multi-variate normal distribution, if $\theta$ are the parameters we need to estimate, $g(\theta)$ = Multivariate Normal $(\hat{\theta}, \hat{\Sigma})$ where the variance covariance matrix is $\hat{\Sigma} = (X'X + \frac{1}{\sigma^2}I)^{-1}$ and the mean $\hat{\theta} = (X'X + \frac{1}{\sigma^2}I)^{-1} + (X'E[Y^*])$

## 1.3 Part C

**Write the steps of the variational Bayes algorithm and program them in R.**
The variational Bayes algorithm is a way to construct an approximate distribution $g(\theta)$ using an expectation procedure, that will minimize the KL divergence in an attempt to approximate a posterior distribution $p(\theta|y)$. The KL divergence is;

$$KL(g||p) = -E_g\Big(\log\Big(\frac{p(\theta|y)}{g(\theta)}\Big)\Big) = -\int \log\Big(\frac{p(\theta|y)}{g(\theta)}\Big)g(\theta)d\theta$$

This expression is minimized when $g \equiv p$ but this is rarely ever the case because we cannot actually summarize $p$ or take posterior draws from it. We write the approximating function as $g(\theta|\phi)$ an the algorithm starts with some guess of $\phi$ and updates it iterative in a way that decreases the KL divergence. At some point $\phi$ stops changing visibly and we stop iterating and use $g(\theta|\phi)$ at the last update to approximate the posterior. The algorithm thus proceeds as follows;

1. Consider a approximating distribution $g(\theta|\phi) = \prod_{i=1}^{n} g_j(\theta_j|\phi_j)$ for the J-dimensional parameter $\theta$

2. Then for every $j$ we examine the expected value of the log posterior density, considering as a function of $\theta_j$ by averaging over the other $J-1$ dimensions of $\theta$.

3. Once the class of approximating distributions $g_j(\theta_j|\phi_j)$ has been identified, we begin with guesses of all hyperparameters $\phi$ and cycle through the distributions $g_j$ updating the vector of hyper-parameters $\phi$ so that log $g_j$ is set to $E_{g_{-j}}(log(p(\theta|y)) = \int logp(\theta|y)g_{-j}(\theta_{-j}|\phi_{-j})d_{\theta_{-j}}$

Following this algorithm will make $g(\theta)$ approach the target posterior distribution $p(\theta|y)$. I generate $\alpha, \beta$ from a multivariate normal distribution such that $[\alpha, \beta] = [0.423, -0.152]$. The variational Bayes algorithm I program estimates the parameters of the posterior density as follows;

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} \sim Normal \begin{bmatrix}0.491 \\ -0.11\end{bmatrix} \begin{bmatrix}0.0154 & 0.0039 \\ 0.0039 & 0.0028\end{bmatrix}$$

# 2 Code

## 2.1 R Code

**Listing 1: R Code**

```
install.packages("msm", dependencies = T)
library(msm)
library(mvtnorm)
################
#Generate data

N <- 100
sd_a <- 1
sd_b <- 1
rho <- 0.05

beta <- rmvnorm(1,
                mean = c(1,0.5),
                sigma = matrix(
                data =c(sd_a^2, sd_a*sd_b*rho,sd_a*sd_b*rho, sd_b^2),
                  nrow = 2))
X <-  matrix( c(rep (1 ,N) , (runif(N, -5, 2) ) ) ,
                   nrow = N, ncol = 2)
z <-  rnorm(N, X\%*\%c (beta ) , 1)
y <-  ifelse( z > 0 , 1 , 0)
#
#Lower bound
lower.bound2<- function(betas, X, stars, sigs, draws){

  ab<- t(X)\%*\%X\%*\%(betas\%*\%t(betas) + sigs)
  part1<- sum(diag(ab))/2
  part2<- t(betas)\%*\%betas + sum(diag(sigs))
  part2<- part2*(beta.mat.prior[1,1])
  part2<- part2/2 + (1/2)*log(det(solve(beta.mat.prior)))
             + (length(betas)/2)*log(2*pi)
  part3<- t(stars)\%*\%stars
  part4<- length(betas)/2 + (1/2)*log(det(sigs))
            + (length(betas)/2)*log(2*pi)
  bounds<-  part1  +  part2 + part3/2 + part4
  parts<- c(-part1, -part2, part3/2, part4)
```

**Listing 2: R Code contd.**

```
  bounds<- list(bounds, parts)
  names(bounds)<- c("bounds","parts")
  return(bounds)
}
#########
##we will use the lower-bound to monitor convergence,
##stopping the model when the lower-bound drops below 1e-8
func.reg<- function(X, draws){
  beta.mat.prior <- diag(1/100, ncol(X))
  beta.VA<- matrix(NA, nrow=1000, ncol=ncol(X))
  ##we begin with random values for the augmented data
  ystars<- rep(0, nrow(X))
  for(j in 1:nrow(X)){
    ystars[j]<- ifelse(draws[j]==1, rtnorm(1, mean=0.5,
    sd=1, lower=0, upper=Inf),
                       rtnorm(1, mean=-0.5, sd=1,
                       lower=-Inf, upper=0) )
  }
  ##we will store the progress of the lower bound on the model
  bounds<- c()
  zz<- 0
  ##this stores the parts of the lower bound
  parts<- matrix(NA, nrow=1000,ncol=4)
  j<- 0
  ##creating a while loop
  while(zz==0){
    j<- j + 1
    ##updating the beta parameters
    beta.VA[j,]<- solve(t(X)\%*\%X + beta.mat.prior)
    \%*\%t(X)\%*\%ystars

    sigs<- solve(t(X)\%*\%X + beta.mat.prior)
    ##computing the inner product of the
    ##covariates current estimates of the coefficients
    stars<- X\%*\%beta.VA[j,]
    denom1<- pnorm(-stars)
    num1<- dnorm(-stars)
    ##now, computing the expected value for each
    ##individual's augmented data, given
    ##current estimates of the approximating
```

**Listing 3: R Code contd.**

```
    ##distribution on the coefficients
    ystars[which(draws==0)]<- stars[draws==0] +
      -num1[which(draws==0)]/denom1[which(draws==0)]
    ystars[which(draws==1)]<- stars[draws==1] +
      num1[which(draws==1)]/(1 - denom1[which(draws==1)])
    ##calculating the lower bound
    trial<- lower.bound2(beta.VA[j,], X, ystars, sigs, draws)
    bounds[j]<- trial$bounds
    parts[j,]<- trial$parts
    if(j>1){
      ##observing convergence
      ab<-abs(bounds[j]-  bounds[j-1])
      if(ab<1e-8){
        zz<-1}
    }
  }
  ##the information to be returned, after convergence
  stuff<- list(bounds, beta.VA[j,], sigs)
  names(stuff)<- c("bound","betas", "sigma")
  return(stuff)
}
example.run <- func.reg(X, y)
```