On bayesian binary probit regression

Feb 16 2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

Model set-up:

Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \in \{0, 1\}$, and we try to model $P(Y_i = 1) = \Phi(\boldsymbol{x}_i^T \boldsymbol{\beta})$

Introduce n latent variables $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)^T$, and that $z_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i, \epsilon_i \overset{iid}{\sim} N(0, 1^2)$

Then we have:

$$Y_i = \begin{cases} 1, & if\ z_i > 0 \\ 0, & if\ z_i \leq 0 \end{cases}$$

since $P(Y_i = 1) = P(z_i > 0) = P(\boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i > 0)$
$= P(\epsilon_i > -\boldsymbol{x}_i^T \boldsymbol{\beta}) = 1 - P(\epsilon_i \leq -\boldsymbol{x}_i^T \boldsymbol{\beta}) = P(\epsilon_i \leq \boldsymbol{x}_i^T \boldsymbol{\beta}) = \Phi(\boldsymbol{x}_i^T \boldsymbol{\beta})$

Thus, the unknown parameters of interest are $(\boldsymbol{\beta}, \boldsymbol{Z})$, and
$y_i \sim bernoulli(\Phi(\boldsymbol{x}_i^T \boldsymbol{\beta})) = bernoulli(P(z_i > 0)), \boldsymbol{Z} \sim N_n(\boldsymbol{x}_i^T \boldsymbol{\beta}, I_n), \boldsymbol{\beta} \sim N_p(\boldsymbol{\beta}_0, \Sigma_0)$.

The joint posterior density of $(\boldsymbol{\beta}, \boldsymbol{Z})$ is:

$p(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{y}, X) \propto p(\boldsymbol{\beta}, \boldsymbol{Z}, \boldsymbol{y}|X) = p(\boldsymbol{y}|\boldsymbol{Z}, \boldsymbol{\beta}, X)p(\boldsymbol{Z}|\boldsymbol{\beta}, X)p(\boldsymbol{\beta}|X)$
$= p(\boldsymbol{y}|\boldsymbol{Z})p(\boldsymbol{Z}|\boldsymbol{\beta}, X)p(\boldsymbol{\beta})$
$= \Pi_{i=1}^n \{P(z_i > 0)^{y_i}[1 - P(z_i > 0)]^{1-y_i}\}|2\pi I_n|^{-\frac{1}{2}} exp(-\frac{1}{2}(\boldsymbol{z} - X\boldsymbol{\beta})^T(\boldsymbol{z} - X\boldsymbol{\beta}))|2\pi\Sigma_0|^{-\frac{1}{2}} exp(-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^T\Sigma_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)) - (1)$

From (1) we can derive the full conditionals for $[\boldsymbol{\beta}|\boldsymbol{z}, X, \boldsymbol{y}]$, $[\boldsymbol{Z}|\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{y}]$:

$p(\boldsymbol{\beta}|\boldsymbol{z}, X, \boldsymbol{y}) \propto p(\boldsymbol{\beta}, \boldsymbol{Z}, \boldsymbol{y}|X)$
$\propto exp(-\frac{1}{2}[(\boldsymbol{\beta}^T X^T - \boldsymbol{z}^T)(X\boldsymbol{\beta} - \boldsymbol{z}) + (\boldsymbol{\beta}^T - \boldsymbol{\beta}_0^T)\Sigma_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)])$
$= exp(-\frac{1}{2}[\boldsymbol{\beta}^T(\Sigma_0^{-1} + X^T X)\boldsymbol{\beta} - 2\boldsymbol{\beta}^T(\Sigma_0^{-1}\boldsymbol{\beta}_0 + X^T\boldsymbol{z}) + \boldsymbol{z}^T\boldsymbol{z} + \boldsymbol{\beta}_0^T\Sigma_0^{-1}\boldsymbol{\beta}_0]$

Let $Q_\beta = \Sigma_0^{-1} + X^T X, l_\beta = \Sigma_0^{-1}\boldsymbol{\beta}_0 + X^T\boldsymbol{z}$,

it can be recognized that $[\boldsymbol{\beta}|\boldsymbol{z}, X, \boldsymbol{y}] \sim N_p(Q_\beta^{-1} l_\beta, Q_\beta^{-1})$

$p(\boldsymbol{Z}|\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{\beta}, \boldsymbol{Z}, \boldsymbol{y}|X)$
$\propto p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{\beta}, X)$
$\propto \Pi_{i=1}^n \{P(z_i > 0)^{y_i}[1 - P(z_i > 0)]^{1-y_i}\}exp(-\frac{1}{2}(\boldsymbol{z} - X\boldsymbol{\beta})^T(\boldsymbol{z} - X\boldsymbol{\beta}))$

where $p(y_i|z_i) = I(y_i = 1)I(z_i > 0) + I(y_i = 0)I(z_i \leq 0)$

$$[z_i|\boldsymbol{\beta}, y_i, X] \sim \begin{cases} N(z_i \in (0, +\infty); \boldsymbol{x}_i^T\boldsymbol{\beta}, 1), & if\ y_i = 1 \\ N(z_i \in (-\infty, 0); \boldsymbol{x}_i^T\boldsymbol{\beta}, 1), & if\ y_i = 0 \end{cases}$$

Implementation

```
set.seed(20740)
# sample size
n= 100
# dim of reg. parameters
p= 2
x= runif(n= n, min= 0, max= 1)

xb= -2+ 6*x

calcProb= function(x) {
    return(exp(xb)/(1+ exp(xb)))
}
pi.x= calcProb(x= xb)

set.seed(21218)
y= rbinom(n= n, size= 1, prob= pi.x)

fit= glm(y ~ x, family= "binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3508  -0.6465   0.2379   0.5827   2.1486
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.3435     0.5653  -4.145 3.39e-05 ***
## x             6.7426     1.3273   5.080 3.78e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 131.791  on 99  degrees of freedom
## Residual deviance:  83.121  on 98  degrees of freedom
## AIC: 87.121
##
## Number of Fisher Scoring iterations: 5
```

```
n1= sum(y)  # Number of successes
n0= n - n1  # Number of failures

X= matrix(c(rep(1, n), x), ncol= p)

require("mvtnorm")
```

```
## Loading required package: mvtnorm
```

```
require("truncnorm")
```

```
## Loading required package: truncnorm
```

```r
# Conjugate prior on the coefficients beta ~ N(beta_0, Q_0)
beta.0= rep(0, p)
Q.0= diag(10, p)

# Initialize parameters
beta= rep(0, p)
z= rep(0, n)

# Number of simulations for Gibbs sampler
S= 15000
# Burn in period
nBurnin= 5000
# Matrix storing samples of the \theta parameter
post.beta= matrix(NA, nrow = S, ncol = p)


# ---------------------------------
# Gibbs sampling algorithm
# ---------------------------------
# Compute posterior variance of beta
# X'X,crossprod(X, X))= t(X)%*% X
Q.beta= t(X)%*% X+ solve(Q.0)
Q.beta.inv= solve(Q.beta)

set.seed(21218)
for (s in 1: S) {
    # Update Mean of z
    mu_z= X %*% beta
    # Draw latent variable z from its full conditional: z | \beta, y, X
    z[y == 0]= rtruncnorm(n0, mean = mu_z[y == 0], sd = 1, a = -Inf, b = 0)
    z[y == 1]= rtruncnorm(n1, mean = mu_z[y == 1], sd = 1, a = 0, b = Inf)

    # Compute posterior mean of theta
    M = Q.beta.inv %*% (t(X) %*% z+ solve(Q.0)%*% beta.0)
    # Draw variable \theta from its full conditional: \theta | z, X
    beta= c(rmvnorm(n= 1, mean= M, sigma= Q.beta.inv))

    # Store the \theta draws
    post.beta[s, ] = beta
}

# ---------------------------
# Get posterior mean of \theta
# ---------------------------
postMean= colMeans(post.beta[-(1: nBurnin), ]); postMean
```

```
## [1] -1.293848  3.761639
```

```r
fit= glm(y ~ x, family = binomial(link = probit))
summary(fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial(link = probit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3627  -0.6685   0.2062   0.6057   2.1345
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.3478     0.3093  -4.357 1.32e-05 ***
## x             3.8669     0.6851   5.645 1.66e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 131.791  on 99  degrees of freedom
## Residual deviance:  83.097  on 98  degrees of freedom
## AIC: 87.097
##
## Number of Fisher Scoring iterations: 5
```

```
mle.beta= fit$coefficients; mle.beta
```

```
## (Intercept)           x
##    -1.34779     3.86694
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.