

Computational Statistics II

Unit C.2: Data augmentation for probit and logit models

Tommaso Rigon

University of Milano-Bicocca

Ph.D. in Economics, Statistics and Data Science



Unit C.2

Main concepts

- Albert & Chib data augmentation for probit models;
 - Pólya-gamma data augmentation for logit models;
 - EM and MM algorithms for logit models.
-
- Associated **R** code is available on the website of the course
 - Additional **R** code (EM tutorial): <https://github.com/tommasorigon/logisticVB>

Main references

- Albert, J. H., & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *JASA*, **88**(422), 669–679.
- Durante, D., & Rigon, T. (2019). Conditionally conjugate mean-field variational Bayes for logistic models. *Statistical Science*, **34**(3), 472–485.
- Polson, N. G., Scott, J. G., & Windle, J. (2013). Bayesian inference for logistic models using Pólya-Gamma latent variables. *JASA*, **108**(504), 1339–1349.

Probit and logit regression models (recap)

- One of the first data augmentation success stories, within the Bayesian framework, is the highly influential Albert & Chib (1993) paper for probit regression.
- Although this approach is nowadays **sub-optimal** in several contexts, it is worth recalling it for historical purposes.
- Let $\mathbf{y} = (y_1, \dots, y_n)^\top$ be a vector of the observed **binary responses**.
- Let \mathbf{X} be the corresponding **design matrix** whose generic row is $\mathbf{x}_i = (1, x_{i2}, \dots, x_{ip})^\top$, for $i = 1, \dots, n$.
- We consider a generalized linear model such that

$$(y_i \mid \pi_i) \stackrel{\text{ind}}{\sim} \text{Bern}(\pi_i), \quad \pi_i = g(\eta_i), \quad \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta} = \beta_1 x_{i1} + \dots + \beta_p x_{ip},$$

where $g(\cdot)$ is either the inverse logit transform or the cdf of a standard normal $\Phi(\cdot)$.

Probit data-augmentation

- The **likelihood** function of a probit regression model is the following

$$\begin{aligned}\pi(\mathbf{y} \mid \beta) &= \prod_{i=1}^n \Phi(\mathbf{x}_i^\top \beta)^{y_i} \{1 - \Phi(\mathbf{x}_i^\top \beta)\}^{1-y_i} \\ &= \prod_{i=1}^n [\mathbb{1}(y_i = 1)\Phi(\mathbf{x}_i^\top \beta) + \mathbb{1}(y_i = 0)\{1 - \Phi(\mathbf{x}_i^\top \beta)\}].\end{aligned}$$

- Let us assume a multivariate Gaussian prior $\pi(\beta)$, leading to the posterior

$$\pi(\beta \mid \mathbf{y}) = \frac{\pi(\beta) \prod_{i=1}^n \Phi(\mathbf{x}_i^\top \beta)^{y_i} \{1 - \Phi(\mathbf{x}_i^\top \beta)\}^{1-y_i}}{\int_{\mathbb{R}^d} \pi(\beta) \prod_{i=1}^n \Phi(\mathbf{x}_i^\top \beta)^{y_i} \{1 - \Phi(\mathbf{x}_i^\top \beta)\}^{1-y_i} d\beta},$$

whose normalizing constant is often (but not always!) **hard to approximate**.

- Whenever computations of the normalizing constant are numerically unstable, we may seek for a suitable data augmentation strategy that enables Gibbs-sampling and EM.

Probit data-augmentation

- We introduce a vector of latent variables $\mathbf{z} = (z_1, \dots, z_n)^\top$ taking values $z_i \in \mathbb{R}$.
- Let us consider the following generative mechanism:

$$z_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \quad \epsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1), \quad i = 1, \dots, n.$$

and transform the scores into binary variables $y_i = \mathbb{1}(z_i > 0)$, for $i = 1, \dots, n$.

- The augmented likelihood of therefore is given by

$$\pi(\mathbf{y}, \mathbf{z} \mid \boldsymbol{\beta}) = \prod_{i=1}^n \phi(z_i \mid \mathbf{x}_i^\top \boldsymbol{\beta}, 1) \{ \mathbb{1}(z_i > 0) \mathbb{1}(y_i = 1) + \mathbb{1}(z_i \leq 0) \mathbb{1}(y_i = 0) \}.$$

- **Exercise.** Prove that the marginal distribution of $\pi(\mathbf{y}, \mathbf{z} \mid \boldsymbol{\beta})$ coincides with $\pi(\mathbf{y} \mid \boldsymbol{\beta})$.

Gibbs sampling for probit models

- It is easy to show (try that as an **exercise**) that the full conditional distribution can be obtained in closed form and they can be easily simulated.
- The full conditional distribution of β is **conjugate** under a Gaussian prior $\beta \sim N(\mathbf{b}, \mathbf{B})$, so that

$$(\beta \mid \mathbf{y}, \mathbf{z}) \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}(\mathbf{X}^\top \mathbf{z} + \mathbf{B}^{-1} \mathbf{b}), \quad \boldsymbol{\Sigma} = (\mathbf{X}^\top \mathbf{X} + \mathbf{B}^{-1})^{-1}.$$

- The elements of the full conditional distribution $(\mathbf{z} \mid \mathbf{y}, \beta)$ are independent and having density

$$\pi(z_i \mid y_i = 1, \beta) \propto \phi(z_i \mid \mathbf{x}_i^\top \beta, 1) \mathbb{1}(z_i > 0)$$

and

$$\pi(z_i \mid y_i = 0, \beta) \propto \phi(z_i \mid \mathbf{x}_i^\top \beta, 1) \mathbb{1}(z_i \leq 0).$$

In other words, the z_i 's follow a **truncated normal distribution**.

- **Homework 2**. Implement this Gibbs sampling using the Pima indian dataset.

Yes, but what about skew-normals?

- Recently, it has been recognized that the “intractable” posterior density

$$\pi(\beta \mid \mathbf{y}) = \frac{\pi(\beta) \prod_{i=1}^n \Phi(\mathbf{x}_i^\top \beta)^{y_i} \{1 - \Phi(\mathbf{x}_i^\top \beta)\}^{1-y_i}}{\int_{\mathbb{R}^d} \pi(\beta) \prod_{i=1}^n \Phi(\mathbf{x}_i^\top \beta)^{y_i} \{1 - \Phi(\mathbf{x}_i^\top \beta)\}^{1-y_i} d\beta},$$

is actually a **known distribution**!

- Indeed, the distribution of $(\beta \mid \mathbf{y})$ is a **unified skew normal** (SUN).
- Do we still need data-augmentation steps? Depending on the context, iid sampling from a SUN distribution is relatively easy (large p) or problematic (large n).

Reference

- Durante, D. (2019). Conjugate Bayes for probit regression via unified skew-normal distributions. *Biometrika*, 106(4), 765–779.

The logit regression model

- The logit regression model is often termed the **canonical choice** for binary regression; see e.g. the classic monograph by McCullagh and Nelder (1986).
- The first key reason is its improved **interpretability**, as the regression coefficients β can be nicely interpreted as log-odds ratios.
- The second reason is its analytical tractability, since the logit case is an **exponential family** of distributions.
- The latter property have a large number of implications, both within the frequentist and Bayesian framework; see e.g. the classic paper by Diaconis and Ylvisaker (1979).
- The **likelihood** function of a logit regression model is the following

$$\pi(\mathbf{y} \mid \beta) = \prod_{i=1}^n \frac{\exp(y_i \mathbf{x}_i^T \beta)}{1 + \exp(\mathbf{x}_i^T \beta)}.$$

The Pólya-gamma distribution

- In a relatively recent paper, Polson et al. (2013) described a data-augmentation scheme for logistic regression based on the Pólya-gamma distribution.
- **Definition (Pólya-gamma).** A positive random variable Z has a Pólya-gamma distribution with parameters $\alpha > 0$ and $\beta \in \mathbb{R}$ denoted as $Z \sim \text{PG}(\alpha, \gamma)$, if

$$Z \stackrel{d}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{G_k}{(k - 1/2)^2 + \gamma^2/(4\pi^2)},$$

where $G_k \sim \text{GA}(\alpha, 1)$ are independent random variables.

- **Remark.** The density $\pi(z \mid \alpha, \gamma)$ of a Pólya-gamma random variable $Z \sim \text{PG}(\alpha, \gamma)$ is expressed in terms of an infinite summation, but it can be **easily simulated**.
- In **R** this can be done using the `rpg.devroye` function of the **BayesLogit** **R** package.

Technical results about the Pólya-gamma distribution

- **Technical lemma 1.** The Laplace transform characterizing the law of $V \sim \text{PG}(\alpha, 0)$ for any $\lambda > 0$ is readily available as

$$\mathbb{E}\{\exp(-\lambda V)\} = \prod_{k=1}^{\infty} \left(1 + \frac{\lambda}{2\pi^2(k - 1/2)^2}\right)^{-\alpha} = \cosh(\sqrt{\lambda/2})^{-\alpha}.$$

- **Technical lemma 2.** The general family of distributions $Z \sim \text{PG}(\alpha, \gamma)$ is generated through the exponential tilting of $V \sim \text{PG}(\alpha, 0)$, since

$$\pi(z \mid \alpha, \gamma) = \frac{e^{-z\gamma^2/2} \pi(z \mid \alpha, 0)}{\mathbb{E}\{\exp(-\gamma^2/2)\}} = \cosh(\gamma/2)^\alpha e^{-z\gamma^2/2} \pi(z \mid \alpha, 0).$$

This can be again proved using the Laplace transform and appealing to the Weierstrass factorization theorem.

- **Intriguing idea?** The Pólya-gamma is also infinitely divisible \implies CRM / NRM can be constructed. The characterizing Lévy-intensity is available as an infinite series.

The data augmentation

- Let $\mathbf{z} = (z_1, \dots, z_n)^\top$ be a vector of latent iid random variables following a $\text{PG}(1, 0)$.
- Then, we define the following **augmented likelihood**

$$\pi(\mathbf{y}, \mathbf{z} \mid \beta) = \prod_{i=1}^n \frac{1}{2} \pi(z_i \mid 1, 0) \exp\{(y_i - 1/2)\mathbf{x}_i^\top \beta - z_i(\mathbf{x}_i^\top \beta)^2/2\}.$$

- Thanks to the **technical lemma 1**, we immediately recognize that this is a valid data augmentation, since

$$\pi(\mathbf{y} \mid \beta) = \int_{\mathbb{R}^n} \pi(\mathbf{y}, \mathbf{z} \mid \beta) d\mathbf{z}.$$

- The augmented log-likelihood is **quadratic** in β , as in the probit case, therefore facilitating posterior computations.

Gibbs sampling for logit models

- The full conditional distribution of β is **conjugate** under a Gaussian prior $\beta \sim N(\mathbf{b}, \mathbf{B})$, so that

$$(\beta \mid \mathbf{y}, \mathbf{z}) \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}\{\mathbf{X}^\top(\mathbf{y} - \mathbf{1}/2) + \mathbf{B}^{-1}\mathbf{b}\}, \quad \boldsymbol{\Sigma} = (\mathbf{X}^\top \mathbf{Z} \mathbf{X} + \mathbf{B}^{-1})^{-1},$$

where $\mathbf{Z} = \text{diag}(z_1, \dots, z_n)$.

- Using the **technical lemma 2**, we recognize that the elements of the full conditional distribution $(\mathbf{z} \mid \mathbf{y}, \beta)$ are independent and such that

$$(z_i \mid \mathbf{y}, \beta) \sim \text{PG}(1, \mathbf{x}_i^\top \beta), \quad i = 1, \dots, n.$$

Note that z_i is independent on \mathbf{y} given β .

- This enables a straightforward Gibbs sampling strategy, provided we can efficiently sample the Pólya-gamma random variables.
- Improved strategies can be devised if $y_i \sim \text{Bin}(n_i, \pi_i)$.

R implementation

```
logit_Gibbs <- function(R, burn_in, y, X, B, b) {  
  p <- ncol(X); n <- nrow(X)  
  out <- matrix(0, R, p) # Initialize an empty matrix to store the values  
  P <- solve(B) # Prior precision matrix  
  Pb <- P %*% b; Xy <- crossprod(X, y - 1 / 2) # Terms appearing in the Gibbs sampling  
  beta <- rep(0, p) # Initialization  
  
  # Gibbs sampling  
  for (r in 1:(R + burn_in)) {  
    eta <- c(X %*% beta)  
    omega <- rpg.devroye(num = n, h = 1, z = eta) # Sampling the Pólya-gamma latent variables  
  
    eig <- eigen(crossprod(X * sqrt(omega)) + P, symmetric = TRUE)  
    Sigma <- crossprod(t(eig$vectors) / sqrt(eig$values))  
    mu <- Sigma %*% (Xy + Pb)  
    A1 <- t(eig$vectors) / sqrt(eig$values)  
    beta <- mu + c(matrix(rnorm(1 * p), 1, p) %*% A1) # Sampling beta  
  
    if (r > burn_in) {  
      out[r - burn_in, ] <- beta # Store the values after the burn-in period  
    }  
  }  
  out  
}
```

The Pima indian dataset

- We consider once again the Pima indian dataset example of **unit B.2**.
- The Pólya-gamma Gibbs sampler has excellent mixing and it requires **no tuning**.

```
# Running the MCMC (R = 30000, burn_in = 5000)
fit_MCMC <- as.mcmc(logit_Gibbs(R, burn_in, y, X, B, b))

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  10018  13592  15411  15182  17369  18900

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  1.587  1.728  1.950  2.051  2.209  2.995

summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#      1      1      1      1      1      1
```

The Newton-Raphson algorithm (recap)

- The Pólya-gamma data augmentation is also useful also for **maximization** purposes.
- For the sake of clarity, let us focus on the MLE, which is defined as

$$\hat{\beta} = \arg \max_{\beta \in \mathbb{R}^p} \left[\sum_{i=1}^n y_i (\mathbf{x}_i^\top \beta) - \log\{1 + \exp(\mathbf{x}_i^\top \beta)\} \right].$$

- Extensions to the MAP case (penalized MLE) is often straightforward.
- The textbook approach for this problem is the **Newton-Raphson** method, which gives

$$\beta^{(r+1)} = \beta^{(r)} + (\mathbf{X}^\top \mathbf{H}^{(r)} \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\pi}^{(r)}),$$

with $\mathbf{H}^{(r)} = \text{diag}\{\pi_1^{(r)}(1 - \pi_1^{(r)}), \dots, \pi_n^{(r)}(1 - \pi_n^{(r)})\}$.

Potential pitfalls

- The Newton-Raphson iterative scheme does not guarantee a monotonic sequence, implying that the algorithm could **fail**.

```
y <- c(rep(0,50), 1, rep(0,50), 0, rep(0, 5), rep(1, 10))      # Binary outcomes
X <- cbind(1, c(rep(0, 50), 0, rep(0.001, 50), 100, rep(-1, 15))) # Design matrix
```

- The MLE is $\hat{\beta} = (-4.603, -5.296)$ and $\log \pi(\mathbf{y} \mid \hat{\beta}) = -15.156$.

- However, the `glm` **R** command, which makes use of Newton-Raphson, does not reach the correct value and raises a warning.

```
coef(glm(y ~ X[, -1], family = "binomial")) # Estimation using Newton-Raphson.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
##      (Intercept)      X[, -1]
## -3.372166e+15 -2.085057e+13
```

The EM algorithm for logistic regression

- An EM strategy automatically leads to a much higher numerical stability, due to the monotonic property.
- **Expectation step.** In first place, note that

$$Q(\beta \mid \beta^{(r)}) = \sum_{i=1}^n (y_i - 1/2) \mathbf{x}_i^\top \beta - \frac{1}{2} \mathbb{E}(z_i) (\mathbf{x}_i^\top \beta)^2 + \text{const},$$

where the expectation is taken w.r.t. Pólya-gamma density, $\pi(z_i \mid \mathbf{y}, \beta^{(r)})$ whose expectation is known:

$$\mathbb{E}(z_i) = \hat{z}_i^{(r)} = \frac{\tanh(\mathbf{x}_i^\top \beta^{(r)} / 2)}{2 \mathbf{x}_i^\top \beta^{(r)}}.$$

- **Maximization step.** Hence, we aim at maximizing $Q(\beta \mid \beta^{(r)})$, obtaining

$$\beta^{(r+1)} = \arg \max_{\beta \in \mathbb{R}^p} Q(\beta \mid \beta^{(r)}) = (\mathbf{X}^\top \hat{\mathbf{Z}}^{(r)} \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - 1/2),$$

where $\hat{\mathbf{Z}}^{(r)} = \text{diag}(\hat{z}_1^{(r)}, \dots, \hat{z}_n^{(r)})$.

The EM for logistic regression

- It turns out that the Pólya-gamma data augmentation not only leads to a stable algorithm, but it has also a **sharp connection** with the Newton-Rapson method.
- With some algebraic manipulation, we can show that

$$\beta^{(r+1)} = (\mathbf{X}^\top \hat{\mathbf{Z}}^{(r)} \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - 1/2) = \beta^{(r)} + (\mathbf{X}^\top \hat{\mathbf{Z}}^{(r)} \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\pi}^{(r)}),$$

- In other terms, the Pólya-gamma EM coincides with a Newton-Rapshon step, having replaced the diagonal matrix $\mathbf{H}^{(r)}$ with $\hat{\mathbf{Z}}^{(r)}$.
- Interestingly, the following inequalities hold true

$$\pi_i^{(r)}(1 - \pi_i^{(r)}) \leq \frac{\tanh(\mathbf{x}_i^\top \boldsymbol{\beta}^{(r)}/2)}{2\mathbf{x}_i^\top \boldsymbol{\beta}^{(r)}} \leq \frac{1}{4}.$$

- The first inequality implies that the EM will perform **smaller but safer** steps compared to the Newton-Rapshon algorithm.

Implementation in R

```
logit_EM <- function(X, y, tol = 1e-16, beta_start = NULL, maxiter = 10000) {  
  
  # Initialization  
  loglik <- numeric(maxiter)  
  Xy <- crossprod(X, y - 0.5)  
  eta <- c(X %*% beta)  
  w <- tanh(eta / 2) / (2 * eta); w[is.nan(w)] <- 0.25  
  loglik[1] <- sum(y * eta - log(1 + exp(eta)))  
  
  # Iterative procedure  
  for (t in 2:maxiter) {  
    beta <- solve(qr(crossprod(X * w, X)), Xy)  
    eta <- c(X %*% beta)  
    w <- tanh(eta / 2) / (2 * eta); w[is.nan(w)] <- 0.25  
  
    loglik[t] <- sum(y * eta - log(1 + exp(eta)))  
    if (loglik[t] - loglik[t - 1] < tol) {  
      return(list(beta = beta, Convergence = cbind(Iteration = (1:t) - 1,  
        Loglikelihood = loglik[1:t])))  
    }  
  }  
  stop("The algorithm has not reached convergence")  
}
```

Solving the pitfalls of Newton-Raphson

- We compare the value of $\log \pi(\mathbf{y} \mid \beta^{(r)})$ obtained through the two algorithms and using the previously considered dataset.
- Both the EM and Newton-Raphson are initialized at $\beta^{(0)} = (0, 0)$. Moreover, this means that $\pi_i^{(0)}(1 - \pi_i^{(0)}) = 1/4$, implying that the first iteration will coincide.

Iteration	0	1	2	3	4	5
Newton-Raphson	-81.098	-38.814	-36.271	-35.433	-26.314	-733.671
EM	-81.098	-38.814	-36.778	-36.332	-36.168	-36.064

- At the 5th iteration, the Newton-Raphson **diverges**, leading to a failure. Conversely, the EM slowly yet steadily increases the log-likelihood.

The MM algorithm for logistic regression

- We finally consider a MM algorithm for finding the MLE of a logistic regression, which is based on the following **minorize** function

$$g(\beta \mid \beta^{(r)}) = \log \pi(\mathbf{y} \mid \beta^{(r)}) + (\mathbf{y} - \boldsymbol{\pi}^{(r)})^\top \mathbf{X}(\beta - \beta^{(r)}) + \frac{1}{2}(\beta - \beta^{(r)})^\top \mathbf{W}(\beta - \beta^{(r)}),$$

with $\mathbf{W} = 0.25\mathbf{X}^\top \mathbf{X}$.

- The minorize function indeed satisfies $g(\beta \mid \beta^{(r)}) \leq \log \pi(\mathbf{y} \mid \beta)$.
- The maximization of leads to the following MM **monotonic** iterative procedure

$$\beta^{(r+1)} = \beta^{(r)} + 4(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\pi}^{(r)}).$$

- This leads to an algorithm that makes even smaller steps than the EM, but it has the advantage of not requiring a matrix inversion at every iteration.

Implementation in R

```
loglik_MM <- function(X, y, tol = 1e-16, beta_start = NULL, maxiter = 10000) {  
  
  # Initialization  
  loglik <- numeric(maxiter)  
  B <- 4 * solve(crossprod(X)) # Bohning and Lindsay matrix  
  eta <- c(X %*% beta)  
  prob <- 1 / (1 + exp(- eta))  
  loglik[1] <- sum(y * eta - log(1 + exp(eta)))  
  
  # Iterative procedure  
  for (t in 2:maxiter) {  
    beta <- beta + B %*% crossprod(X, y - prob)  
    eta <- c(X %*% beta)  
    prob <- 1 / (1 + exp(- eta))  
    loglik[t] <- sum(y * eta - log(1 + exp(eta)))  
    if (loglik[t] - loglik[t - 1] < tol) {  
      return(list(beta = beta, Convergence = cbind(Iteration = (1:t) - 1,  
        Loglikelihood = loglik[1:t])))  
    }  
  }  
  stop("The algorithm has not reached convergence")  
}
```

EM and MM comparison (simulated data)

