Inference and Estimation in GLMMs

Recall GLMMs

Conditional on cluster specific random effects $\boldsymbol{b}_i$,

$$\begin{aligned} E(Y_{ij}|b_i) &= \mu_{ij}^{\mathbf{b}} \\ var(Y_{ij}|b_i) &= \phi a_{ij}^{-1} v(\mu_{ij}^{\mathbf{b}}) \end{aligned}$$

Random Effects Model :

$$g(\mu_{ij}^{\mathbf{b}}) = \mathbf{X}_{ij}^T \beta + \mathbf{Z}_{ij}^T \mathbf{b}_i$$

where

$$\mathbf{b}_i \sim N\{0, \mathbf{D}_0(\boldsymbol{\theta})\}$$

Likelihood

$$e^{\ell(\beta,\boldsymbol{\theta})} \propto |\mathbf{D}|^{-\frac{1}{2}} \int exp^{\{\sum_{i=1}^{n} \ell_i(Y_i|\mathbf{b};\beta) - \frac{1}{2}\mathbf{b}^{\mathbf{T}}\mathbf{D}^{-1}\mathbf{b}\}} d\mathbf{b}$$

How do we do estimation?

$$e^{\ell(\beta,\boldsymbol{\theta})} \propto |\mathbf{D}|^{-\frac{1}{2}} \int exp^{\{\sum_{i=1}^{n} \ell_i(Y_i|\mathbf{b};\beta) - \frac{1}{2}\mathbf{b}^{\mathbf{T}}\mathbf{D}^{-\mathbf{1}}\mathbf{b}\}} d\mathbf{b}$$

Main challenge: Difficult to take that integral!

Conditional Inference

Recall: this is potential approach for dealing with random effects

- Idea: Calculate the sufficient statistic for $\mathbf{b}_i$s and make inference using the conditional likelihood on the sufficient statistics.
- Pro: robustness (no distributional assumptions on $\mathbf{b}_i$) and likelihood has closed form
- Con: only works in special cases (e.g. logistic GLMM and poisson log linear GLMM)
- Con: you've lost the random effects

Instead: can we just take the MLE?

Full MLE Approach

$$e^{\ell(\beta,\boldsymbol{\theta})} \propto |\mathbf{D}|^{-\frac{1}{2}} \int exp^{\{\sum_{i=1}^{n} \ell_i(Y_i|\mathbf{b};\beta) - \frac{1}{2}\mathbf{b}^{\mathbf{T}}\mathbf{D}^{-\mathbf{1}}\mathbf{b}\}} d\mathbf{b}$$

Idea:

To directly evaluate the integrated loglikelihood$\ell(\beta, \boldsymbol{\theta})$ using numerical integration methods, such as adaptive Gaussian quadratures or Monte-Carlo.

Numerical Integration: Adaptive Gaussian Quadrature

Idea: To numerically evaluate the integral in $\ell(\boldsymbol{\beta}, \boldsymbol{\theta})$ using a weighted sum over predefined set of quadrature points, i.e., by replacing the integral by a sum.

Advantages:

- Full MLE if the integral can be evaluated precisely using sufficient number of quadrature points.
- For a one-dimensional integral, e.g., random intercept models, 10-20 quadrature points give a very precise evaluation of the integral and calculations are very quick.

Adaptive Gaussian Quadrature: (2)

Disadvantages:

- It is only feasible if the dimension of integral is small, e.g., in simple random effect models for longitudinal/clustered data.
- For a multi-dimensional integral, the number of quadrature points grows exponentially and such a direct quadrature approximation can be prohibited, e.g., for a 5-dimensional integral, the number of quadratures is 20^5.
- For spatial data, the dimension of integration is the sample size. Direct numerical integration is infeasible.

References:

Golub and Welsch (1969); Pinheiro and Bates (1995)

Adaptive Gaussian Quadrature: Operationally

Quadrature replaces

$$I = \int w(u)g(u)du \text{ by } \widetilde{I} = \sum_{i=1}^{n} w_i g(u_i)$$

for nodes u_i and weights w_i (note n is not sample size)

- For GLMM: we have integrals w.r.t. a Gaussian density
- Gauss-Hermite quadrature is designed for problems of this type

Adaptive Gaussian Quadrature: Operationally (2)

Gauss-Hermite Quadrature:

$$I = \int_{-\infty}^{\infty} \exp(-u^2) g(u) du \approx \sum_{i=1}^{n} w_i g(u_i) = \widetilde{I}$$

where $g(\cdot)$ is a polynomial of degree $2n - 1$ or less.

u_i is the i^{th} of n roots of the Hermite polynomial $H_n(u)$ with weight

$$w_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(u_i)]^2}$$

where $H_n(u)$ are orthogonal polynomial sequence given by

$$H_n(u) = (-1)^n \exp(-u^2) \frac{\partial^n}{\partial u^n} \exp(-u^2)$$

Adaptive Gaussian Quadrature: Getting Weights (3)

R functions to get the weights

```
> library(statmod)
> quad <- gauss.quad(4, kind="hermite")
> round(quad$nodes, 3)
[1] -1.651 -0.525 0.525 1.651
> round(quad$weights, 3)
[1] 0.081 0.805 0.805 0.081
> quad <- gauss.quad(5, kind="hermite")
> round(quad$nodes, 3)
[1] -2.020 -0.959 0.000 0.959 2.020
> round(quad$weights, 3)
[1] 0.020 0.394 0.945 0.394 0.020
```

Adaptive Gaussian Quadrature: Example (3)

$$\int_{-\infty}^{\infty} \exp(-u^2)\exp(cos(u)^2 + sin(u)^3)du$$

```
> u <- seq(-100,100,0.1)
> sum(exp(-u^2)*exp(cos(u)^2+sin(u)^3)*0.1)
[1] 3.829421
> quad <- gauss.quad(5, kind="hermite")
> sum(quad$weights*exp(cos(quad$nodes)^2+sin(quad$nodes)^3
[1] 3.895256
> quad <- gauss.quad(10, kind="hermite")
> sum(quad$weights*exp(cos(quad$nodes)^2+sin(quad$nodes)^3
[1] 3.830933
```

Adaptive Gaussian Quadrature: Multiple Dimensions

Suppose $\boldsymbol{\theta}$ is two dimensional:

$$I = \int f(\boldsymbol{\theta}) d\boldsymbol{\theta} = \int \int f(\theta_1, \theta_2) d\theta_2 d\theta_1 = \int f^*(\theta_1) d\theta_1$$

where $f^*(\theta_1) = \int f(\theta_1, \theta_2) d\theta_2$. Then

$$\widetilde{I} = \sum_{i=1}^{m} w_i f^*(\theta_{1_i}) = \sum_{i=1}^{n_1} w_i \sum_{j=1}^{n_2} w_j f(\theta_{1_i}, \theta_{2_j}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_i w_j f(\theta_{1_i}, \theta_{2_j})$$

which is a Cartesian product.

Question: Where is the "adaptive" part?

Adaptive Gaussian Quadrature

- Note that the nodes are selected without regard to the function $g(\cdot)$
  - Good if $w(u)g(u)$ behaves like a normal
  - But in general, quadrature points may not lie in interesting or important region

- Adaptive: rescale and shift quadrature points s.t. more quadrature points in region of interest

- Adaptive version requires fewer quadrature points, but requires calculating mode of $w(u)g(u)$ which can be slow!

Recall: gets complicated when dimensionality of the integral gets higher $\rightarrow$ Approximations