(Moderately) Advanced Hierarchical Models

Ben Goodrich

StanCon: January 12, 2018
Obligatory Disclosure

• Ben is an employee of Columbia University, which has received several research grants to develop Stan
• Ben is also a manager of GG Statistics LLC, which utilizes Stan for business purposes
• According to Columbia University policy, any such employee who has any equity stake in, a title (such as officer or director) with, or is expected to earn at least $5,000.00 per year from a private company is required to disclose these facts in presentations
Goals for the Tutorial

• Thinking in terms of conditional distributions is good
• But conditional distributions make things harder for NUTS
• By reparameterizing, you can make hierarchical models easier for NUTS
• Also, want to learn about matrix decompositions and priors on components
Cluster Sampling Designs

Consider a more elaborate version of the school example:

\[
\begin{align*}
\tau & \sim \text{Exponential}(r_{\tau}) \\
\alpha & \sim \mathcal{N}(\mu_{\alpha}, \mu_{\beta}) \\
\alpha_j & \sim \mathcal{N}(\alpha, \tau) \quad \forall j \\
\sigma & \sim \text{Exponential}(r_{\sigma}) \\
\sigma_j & \sim \text{Exponential}\left(\frac{1}{\sigma}\right) \quad \forall j \\
\epsilon_{ij} & \sim \mathcal{N}(0, \sigma_j) \quad \forall i \in j \\
\beta & \sim \mathcal{N}(\mu_{\beta}, \sigma_{\beta}) \\
y_{ij} & \equiv \alpha_j + \beta \times \text{class\_size}_i + \epsilon_{ij} \quad \forall i, j
\end{align*}
\]
Frequentist vs. Bayesian Perspective

- The previous DGP seems reasonable but
  - In order to estimate $\alpha$, $\beta$, and $\sigma$ consistently as $J \uparrow \infty$, $\alpha_j$ and $\sigma_j$ must be integrated out of the likelihood function.
  - However, $\sigma_j$ cannot be integrated out of the likelihood function analytically.
  - Therefore, the `lmer` function in `lme4` requires $\sigma_j = \sigma \forall j$.

- Bayesian methods condition on the $J$ groups rather than integrating over the process by which they were selected.
- MCMC methods may have considerable difficulty drawing from this posterior distribution sufficiently efficiently.
- By reparameterizing, you can improve the prospects for Stan to sample from this posterior distribution well.
Sampling Efficiency

• If we could obtain $S$ independent draws from a posterior distribution, posterior means would converge at a $\sqrt{S}$ rate

• But we cannot obtain independent draws from non-trivial posterior distributions

• MCMC methods yield $S$ dependent draws from posterior distributions and posterior means converge at a $\sqrt{S_{\text{eff}}}$ rate

• If the draws are moderately dependent, then $\sqrt{S_{\text{eff}}} \approx \sqrt{S}$ and everything is basically fine

• If the draws are severely dependent, then there is no finite $S$ that yields reliable posterior means

• NUTS produces draws that have less dependence than other MCMC algorithms

• But whether $\sqrt{S_{\text{eff}}} \approx \sqrt{S}$ under NUTS depends on the (parameterization of the) posterior distribution
When Does NUTS “Fail”?

- NUTS only uses first derivatives of the log-posterior kernel
- A curve can be approximated by a line over a small interval
- NUTS would work perfectly with only first derivatives if higher derivatives of a posterior distribution were constant
- Independent Gaussian log-PDFs have constant second derivatives:
  \[
  \frac{\partial^2}{\partial \mu \partial \mu} \left( -\frac{1}{2} \left( \frac{x-\mu}{\sigma} \right)^2 \right) = -\frac{1}{\sigma}
  \]
- When the higher derivatives are not constant, NUTS has to reduce its step size to approximate a curve sufficiently well
- If the higher derivatives change rapidly, the step size can go to zero numerically and NUTS takes infinite steps / time
- By changing the parameterization, you change derivatives without changing the posterior means or inferences
Let’s simplify to the case where only the intercept varies across groups, i.e. \( \alpha_j \sim \mathcal{N} (\alpha, \sigma) \) \( \forall j \)

- \( \sigma = e^\omega \) is unknown and \( \omega \) has an improper uniform prior
- \( \mathcal{N} (\alpha, \sigma) \overset{d}{=} \alpha + \sigma \times \mathcal{N} (0, 1) \) and similarly for other distributions in the location-scale family

You can often help Stan via transformations

\[
\begin{align*}
  u_j & \sim \mathcal{N} (0, 1) \implies \\
  \alpha_j = \alpha + e^\omega u_j \forall j & \sim \mathcal{N} (\alpha, e^\omega)
\end{align*}
\]

- `vector[J] u` would be declared in the parameters block
- `vector[J] alpha` would be declared in the transformed parameters block
- The second derivative with respect to each \( u_j \) is constant
- Look at the bivariate prior for \( \omega \) vs. that of \( u_j, \omega \)
library(rgl)

kernel <- function(alpha, omega) {
  dnorm(alpha, sd = exp(omega), log = TRUE)
}

LIM <- c(-2,2)
persp3d(kernel, xlim = LIM,
       ylim = LIM, zlab = "log kernel")

reparameterized_kernel <- function(u, omega) {
  dnorm(u, log = TRUE)
}
persp3d(reparameterized_kernel, xlim = LIM,
       ylim = LIM, zlab = "log kernel")
Coefficients Depending on Other Coefficients Again
Recall our Stan program where the coefficient on age is a noisy linear function of the person’s income:

```stan
data {
  int<lower=1> N; vector[N] age;
  vector[N] income; int<lower=0,upper=1>[N] vote;
}
parameters {
  vector[2] lambda; // intercept / slope for age's effect
  vector[N] noise; // error in effect of age
  real<lower=0> sigma; // sd of error in beta_age
  vector[2] beta; // intercept / slope for outcome
}
model {
                      + sigma * noise; // non-centering
                  + beta_age .* age;
  target += binomial_logit_lpmf(vote | eta);
  target += normal_lpdf(noise | 0, 1);
} // priors on lambda, sigma, and beta
```
Centered Parameterization

The following is conceptually the same but often problematic:

data {
  int<lower=1> N;  vector[N] age;
  vector[N] income;  int<lower=0,upper=1>[N] vote;
}
parameters {
  vector[2] lambda;  // intercept / slope for age’s effect
  vector[N] beta_age;  // coefficient on age
  real<lower=0> sigma;  // sd of error in beta_age
  vector[2] beta;  // intercept / slope for outcome
}
model {
  + beta_age .* age;
  target += binomial_logit_lpmf(vote | eta);
  target += normal_lpdf(beta_age | lambda[1] +
                        lambda[2] * income, sigma);
}  // priors on lambda, sigma, and beta
Multivariate Matt Trick

- If $\beta_j \sim \text{MultiNormal}(\mu, \Sigma)$, Stan can have difficulty drawing from the joint posterior distribution
  - When $\Sigma_{kk}$ is small, $\beta_{kj}$ must fall in a narrow range, which entails a small stepsize for NUTS
  - When $\Sigma_{kk}$ is large, $\beta_{kj}$ can fall in a wide range, which requires a large stepsize or else many small steps
- You can help Stan with this problem via transformations

$$u_{kj} \sim \text{Normal}(0, 1) \forall k, j \implies \beta_j = \mu + \sigma L u_j \sim \text{MultiNormal}(\mu, \sigma^2 LL^\top)$$

where $\sigma L$ is the Cholesky factor of $\Sigma = \sigma^2 LL^\top$ and $\sigma$ is the standard deviation of the errors

- Both rstanarm and brms do things like this
Decomposing a Covariance Matrix

• Suppose $\beta_j \sim \mathcal{N} (\mu, \Sigma)$ where $\beta_j$ is a $K$-vector for group $j$

• Many people find specifying a prior on the $K \times K$ covariance matrix to be difficult. You will see (inverse) Wishart priors in the literature which are confusing but conjugate with the multivariate normal and thus facilitate Gibbs sampling.

• With Stan, you are free to do what makes sense, such as

\[
\Sigma = \Delta \Lambda \Delta \quad [\text{stds x correlation x stds}]
\]
\[
\Delta^2_k = \tau \pi_k \forall k
\]
\[
\tau = \gamma^2 K
\]
\[
\gamma \sim \text{Jeffreys / Gamma / Exponential}
\]
\[
\pi \sim \text{Dirichlet} (a)
\]
\[
\Lambda \sim \text{prior?}
\]

• $\pi$ is a simplex, so the $k$th variance, $\Delta^2_k$, is a proportion of $\tau$, which is the trace of $\Sigma$ & a function of a scale parameter, $\gamma$
Prior for a Correlation Matrix

• There are many choices for a prior on a scale parameter, such as Jeffreys if you want to be non-informative

• A Dirichlet ($\mathbf{a}$) prior for $\mathbf{\pi}$ is pretty easy to specify, such as $\mathbf{a} = \mathbf{1}$ if you want to be jointly uniform on the $K$-simplex

• There is an easy and possibly non-informative prior for a correlation matrix $\Lambda$, $f(\Lambda | \eta) = \frac{1}{c(\eta, K)} |\Lambda|^{\eta-1}$ called “LKJ”

• $\eta$ acts like the shape parameter of a Beta distribution
  • if $\eta = 1$, $f(\Lambda | \eta) = \frac{1}{c(\eta, K)}$ is constant
  • if $\eta > 1$, $\mathbf{I}$ is the modal correlation matrix and the only correlation matrix with positive density as $\eta \uparrow \infty$
  • if $\eta < 1$, $\mathbf{I}$ is at the trough of the distribution of correlation matrices, which is a weird thing to believe

• But $\Lambda = \mathbf{C} \mathbf{C}^\top$ where $\mathbf{C}$ is a Cholesky factor

• Can specify a prior on $\mathbf{C}$ such that $\Lambda$ has the LKJ prior
A Multivariate Matt Trick with \texttt{brms}

\begin{verbatim}
library(brms)
post <- brm(Reaction ~ Days + (Days | Subject),
            data = lme4::sleepstudy)  # no warnings!

make_stancode(Reaction ~ Days + (Days | Subject),
             data = lme4::sleepstudy)
\end{verbatim}
Data and Transformed Data Blocks

data {
  int<lower=1> N;    // total number of observations
  vector[N] Y;      // response variable
  int<lower=1> K;    // number of population-level effects
  matrix[N, K] X;   // population-level design matrix
  // data for group-level effects of ID 1
  int<lower=1> J_1[N];
  int<lower=1> N_1;
  int<lower=1> M_1;
  vector[N] Z_1_1;
  vector[N] Z_1_2;
  int<lower=1> NC_1;
  int prior_only;    // should the likelihood be ignored?
}

transformed data {
  int Kc = K - 1;
  matrix[N, K - 1] Xc;    // centered version of X
  vector[K - 1] means_X; // column means of X before centering
  for (i in 2:K) {
    means_X[i - 1] = mean(X[, i]);
    Xc[, i - 1] = X[, i] - means_X[i - 1];
  }
}

parameters {
  vector[Kc] b;    // population-level effects
  real temp_Intercept; // temporary intercept
  real<lower=0> sigma; // residual SD
  vector<lower=0>[M_1] sd_1; // group-level standard deviations
  matrix[M_1, N_1] z_1; // unscaled group-level effects
  // cholesky factor of correlation matrix
  cholesky_factor_corr[M_1] L_1;
}

transformed parameters {
  // group-level effects
  matrix[N_1, M_1] r_1 = (diag_pre_multiply(sd_1, L_1) * z_1)';
  vector[N_1] r_1_1 = r_1[, 1];
  vector[N_1] r_1_2 = r_1[, 2];
}

model {
  vector[N] mu = Xc * b + temp_Intercept;
  for (n in 1:N) {
    mu[n] = mu[n] + (r_1_1[J_1[n]]) * Z_1_1[n] + (r_1_2[J_1[n]]) * Z_1_2[n];
  }
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, 288.65, 56);
  target += student_t_lpdf(sigma | 3, 0, 56) - 1 * student_t_lccdf(0 | 3, 0, 56);
  target += student_t_lpdf(sd_1 | 3, 0, 56) - 2 * student_t_lccdf(0 | 3, 0, 56);
  target += lkj_corr_cholesky_lpdf(L_1 | 1);
  target += normal_lpdf(to_vector(z_1) | 0, 1);
  // likelihood including all constants
  if (!prior_only) {
    target += normal_lpdf(Y | mu, sigma);
  }
}

generated quantities {
  // actual population-level intercept
  real b_Intercept = temp_Intercept - dot_product(means_X, b);
  corr_matrix[M_1] Cor_1 = multiply_lower_tri_self_transpose(L_1);
  vector<lower=-1,upper=1>[NC_1] cor_1;
  // take only relevant parts of correlation matrix
  cor_1[1] = Cor_1[1,2];
}
parameters {
  vector[Kc] b;  // population-level effects
  real temp_Intercept;  // temporary intercept
  real<lower=0> sigma;  // residual SD
  vector<lower=0>[M_1] sd_1;  // group-level standard deviations
  matrix[M_1, N_1] z_1;  // unscaled group-level effects
  // cholesky factor of correlation matrix
  cholesky_factor_corr[M_1] L_1;
}
transformed parameters {
  // group-level effects
  matrix[N_1, M_1] r_1 = (diag_pre_multiply(sd_1, L_1) * z_1)';
  vector[N_1] r_1_1 = r_1[, 1];
  vector[N_1] r_1_2 = r_1[, 2];
}
model {
  vector[N] mu = Xc * b + temp_Intercept;
  for (n in 1:N) {
    mu[n] = mu[n] + (r_1_1[J_1[n]]) * Z_1_1[n] + (r_1_2[J_1[n]]) * Z_1_2[n];
  }
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, 288.65, 56);
  target += student_t_lpdf(sigma | 3, 0, 56) - 1 * student_t_lccdf(0 | 3, 0, 56);
  target += student_t_lpdf(sd_1 | 3, 0, 56) - 2 * student_t_lccdf(0 | 3, 0, 56);
  target += lkj_corr_cholesky_lpdf(L_1 | 1);
  target += normal_lpdf(to_vector(z_1) | 0, 1);
  // likelihood including all constants
  if (!prior_only) {
    target += normal_lpdf(Y | mu, sigma);
  }
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = temp_Intercept - dot_product(means_X, b);
  corr_matrix[M_1] Cor_1 = multiply_lower_tri_self_transpose(L_1);
  vector<lower=-1,upper=1>[NC_1] cor_1;
  // take only relevant parts of correlation matrix
  cor_1[1] = Cor_1[1,2];
}
Conclusion

• Should use hierarchical modeling unless there is a strong reason not to
• Hierarchical models are more straightforward from a Bayesian perspective
• NUTS does a better job with hierarchical modeling than does Gibbs
• But the parameterization can make a big difference to NUTS