Fitting multiparameter models

Math 315, Adam Loy

Modeling heights

For a small number of parameters, the grid approximation can be used to approximate the posterior distribution in a multiparameter model. Below, we work through the steps to fit the informative model for heights discussed in class:

$$\begin{aligned}
Y_1, \ldots, Y_n &\overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2) \\
\pi(\mu, \sigma^2) &= \pi(\mu)\pi(\sigma^2) \\
\mu &\sim \mathcal{N}(178,\ 20) \\
\sigma^2 &\sim \text{Unif}(0, 50)
\end{aligned}$$

Work through the steps of this handout carefully, so that you understand both the R code involved and what you are accomplishing with each code chunk.

1. Load the data

To begin, load the data:

```r
adults <- read.csv("http://aloy.rbind.io/data/adults.csv")
```

2. Explore the implications of the prior specification

Next, explore the prior predictive distribution for this setting by

i. simulating $S = 1000$ $\mu^{(i)}$ draws from the prior for μ,
ii. simulating $S = 1000$ ${\sigma^2}^{(i)}$ draws from the prior for σ^2, and
iii. simulating $S = 1000$ $Y^{(i)}$ draws from $\mathcal{N}(\mu^{(i)}, {\sigma^2}^{(i)})$.

Does the prior specification appear reasonable considering the world's tallest person ever was 272 cm tall.

3. Choose a grid that covers the posterior density

To use grid approximation, we need to choose a reasonably large grid to ensure that non-zero density is placed over the region where the posterior will reside. Here, the Cartesian product of $\mu \in [118, 238]$ and $\sigma \in [0.1, 50]$ seems reasonable. In R, the `expand.grid()` command returns a data frame containing all combinations of the two grids:

```r
param_grid <- expand.grid(
  mu = seq(from = 118, to = 238, length.out = 1000),
  sigma = seq(from = 0.1, to = 50, length.out = 1000)
)
```

Check the number of rows and columns, does this match your intuition?

4. Evaluate the log prior density over the grid

Once you have the grid in hand, you can evaluate the joint prior density over the grid, taking advantage of R's vectorized operations. As we have discussed in class, it is wiser to evaluate the prior, likelihood, and posterior on the log scale to avoid rounding error erroneously forcing the posterior probabilities to zero.

```r
logprior <- dnorm(param_grid$mu, 178, 20, log = TRUE) + dunif(param_grid$sigma, 0, 50, log = TRUE)
```

5. Evaluate the log likelihood density over the grid

To evaluate the log likelihood using R's built-in `dnorm()` command, we either need to use a for loop, or to use the `Vectorize()` command to extend `dnorm()` for use with multiple data points. Both code chunks are shown below. Both commands produce the same result, so be sure to understand at least one approach well.

Using a for loop:

```r
# Set up a place to store the log likelihood
loglike <- numeric(length = nrow(param_grid))

# Evaluate the log likelihood over each grid point and store
for(i in 1:nrow(param_grid)) {
  loglike[i] <- sum(dnorm(adults$height, mean = param_grid[i, "mu"],
                          sd = param_grid[i, "sigma"], log = TRUE))
}
```

Vectorizing `dnorm()`:

```r
# Writing a log likelihood function
llnorm <- function(x, mu, sigma) {
  sum(dnorm(x, mean = mu, sd = sigma, log = TRUE))
}

# Vectorize llnorm to use all of the x vector for each mu-sigma combo
llnorm <- Vectorize(llnorm, vectorize.args = c("mu", "sigma"))

# Calculate the log likelihood for each point on grid
loglike <- llnorm(x = adults$height, mu = param_grid$mu, sigma = param_grid$sigma)
```

6. Compute the posterior density via the log posterior

Finally, you can calculate the log posterior and exponentiate it to obtain the posterior distribution.

```r
logposterior <- logprior + loglike
unstd_posterior <- exp(logposterior - max(logposterior)) # numeric stability
posterior <- unstd_posterior / sum(unstd_posterior)
```

Note: `logposterior - max(logposterior)` simply subtracts each value of the log posterior by the maximum of the log posterior, which simply improves the numerical stability (i.e. avoids more rounding error).

7. Use Monte Carlo sampling to draw $(\mu^{(i)}, \sigma^{2^{(i)}})$ pairs from the posterior

Just like in the one-parameter setting, we can use Monte Carlo sampling to draw samples from our grid-approximate posterior distribution. Since we are drawing pairs of parameters rather than single values, I recommend using the `sample_n()` command in the **dplyr** R package so that you can draw samples from the rows of a data frame.

```r
library(dplyr)
posterior_draws <- sample_n(param_grid, size = 1e4, replace = TRUE, weight = posterior)
```

8. Analyze the joint or marginal posteriors

Now, you have the joint posterior distribution in hand, so you can conduct inference as desired. Complete the following steps to analyze the posterior in this setting:

a. Plot the joint posterior density as: (i) a scatterplot where you make the points transparent (say `alpha = 0.5`), and (ii) a contour plot. I would recommend using the **ggplot2** R package for both of these tasks, using either the `geom_point()` or `geom_density2d()`. You can fill in the code below to achieve this:

```r
# Fill in code for a scatterplot
ggplot(data = ___, aes(x = ___, y = ___)) +
  geom_point(alpha = 0.2) +
  labs(x = expression(mu),
       y = expression(sigma))

# Fill in code for a contour plot
ggplot(data = ___, aes(x = ___, y = ___)) +
  geom_density2d() +
  labs(x = expression(mu),
       y = expression(sigma))
```

b. Plot the marginal posterior density of μ, the mean height.

c. Calculate and interpret a 93% credible interval for the mean height in this population.