MCMC Inference for a Bayesian Logistic Regression Model

Alex Franks

Due March 2

A model for bee toxicity

A environmental agency is testing the effects of a pesticide that can cause acute poisoning in bees, the world's most important pollinator of food crops. The environmental agency collects data on exposure to different levels of the pestidicide in parts per million (ppm). The agency also identifies collapsed beehives, which they expect could be due to acute pesticide poisoning. In the data they collect, each observation is a pair (x_i, y_i), where x_i represents the dosage of the pollutant and y_i represents whether or not the hive survived. Take $y_i = 1$ means that the beehive has collapsed from poisoning and $y_i = 0$ means the beehive survived. The agency collects data at 20 different sites, each of which was exposed to a different dosages. In addition, the agency observed "control'' hives for which there was no pesticide detected at all. None of the control hives experienced collapse. The data for the first ten exposed hives are shown below:

x	y
1.06	0
1.41	1
1.85	1
1.50	1
0.46	0
1.21	1
1.25	1
1.09	1
1.76	1
1.75	1

Assume that beehive collapse, y_i, given pollutant exposure level x_i, is $Y_i \sim \text{Bernoulli}(\theta(x_i))$,

where $\theta(x_i)$ is the probability of death given dosage x_i. We will assume the logistic regression model, $\text{logit}(\theta_i(x_i)) = \alpha + \beta x_i$ where $\text{logit}(\theta)$ is defined as $\log(\theta/(1-\theta))$.

1a The dose at which there is a 50% chance of beehvive collapse, $\theta(x_i) = 0.5$, is known as LD50 ("lethal dose 50%"), and is often of interest in toxicology studies. We will use γ to denote the LD50. Solve for γ as a function of α and β.

1b. Environmentalist prefer to specify their prior knowledge in terms of γ and α (as opposed to α and β) due to the interpretability of these parameters. Write the likelihood for the proposed model in terms of α and γ.

1c Environmentalists think a priori that γ is positive and centered around 1ppm. State why it doesn't make sense for γ to be negative and propose a Gamma prior for γ with positive support, centered near 1ppm with appropriate uncertainty. State the parameters of your prior and justify in 1-2 sentences.

1d. State the interpretation for α in terms of hive collapse and toxic exposure. Assume we are certain that the pesticide cannot reduce the probability of beehive collapse. State what the above assumption implies about the prior support for α. Assume an improper uniform prior on the support of α.

1e Write down the posterior density up to a proportionality constant. Fill in the template code below, which should return the log posterior density evaluated at any (α, γ) given the data provided.

As a reminder, the posterior density may have *extremely* small values, especially when we initialize the sampler and may be far from the high posterior mode areas. For example, computing the ratio of a normal density 1000 standard deviations from the mean to a normal density 1001 standard deviations from the mean fails because in both cases `dnorm` evalutes to 0 due to numerical underflow and 0/0 returns NaN. However, we can compute the log ratio of densities:

```
dnorm(1000) / dnorm(1001)
```

```
[1] NaN
```

```
dnorm(1000, log=TRUE) - dnorm(1001, log=TRUE)
```

```
[1] 1000.5
```

Fill in the code below.

```
## Log posterior function.
log_posterior <- function(alpha, gamma) {

  ## Compute the probabilities as a function of alpha and beta
  ## for the observed x, y data
}
```

Implementing the Metropolis-Hastings Algotihm

Let $r = \min(1, \frac{p(\theta^*|y)}{p(\theta_t|y)} \frac{J(\theta_t|\theta^*)}{J(\theta^*|\theta_t)})$. In the accept/reject step of the your implementation of the MH algorithm checking $u < r$ is equivalent to checking whether $log(u) < log(r)$. Completing the accept/reject on the log scale will avoid any underflow issues and prevent our code from crashing.

Complete the Metropolis-Hastings sampler by filling in the missing pieces of the algorithm below. `theta_0` is a vector of length 2, with the first argument as the initial α value and the second argument as the initial γ value. As your proposal, use $J(\theta*|\theta_t) \sim Normal(\theta_t, \Sigma)$. You can sample from the multivariate normal using `mvtnorm::rmvnorm`. The effectiveness of your sampler will be determined by the tuning parameter, Σ, the covariance of the bivariate normal distribution. This determines the size / shape of the proposal. Set Σ using the `cov` argument in your sampler.

Burn in your sampler for 10000 iterations and then save 10000 samples after burnin. Report a 95% credible region for LD50 and report the effective sample size for γ (you can use`coda::effectiveSize` function). See if you can improve upon this effective sample size from your first run by finding a new setting for `cov`. What's the best effective sample size you were able to achieve? *Hint*: make a traceplot of the samples and/or compute the acceptance rate to identify if your proposal variance is too large or too small. We recommend using `bayesplot`. What is the largest effective size for γ that you were able to achieve? Save the effective sample size of γ in a variable called `ld50_ess_mh`.

```
################################################
## Metropolis-Hastings for the Logistic Model
################################################

## Function to generate samples using the Metropolis-Hasting Sampler


## burnin: amount of iterations to discard to reduce dependence on starting point
## samples: total number of samples after burnin
## theta_0: initialization of the form c(alpha_init, beta_init) for some values alpha_init,
## cov: covariance of the Gaussian proposal density
```

```r
mh_logistic <- function(samples, burnin, theta_0=c(1, 1), cov=diag(2)){

    # Initialize parameters.
    theta_t <- theta_0

    ## Create a matrix where we will store samples
    theta_out <- matrix(0, nrow=burnin+samples, ncol=2, dimnames=list(1:(burnin + samples), c

    for(i in 1:(burnin+samples)){

        ## MH Algorithm here

        ## Save the draw
        theta_out[i, ] <- NULL # new theta_t goes here

    }

    ## Chop off the first part of the chain -- this reduces dependence on the starting point
    if(burnin == 0)
      theta_out
    else
      theta_out[-(1:burnin), ]
}

## Report 95% credible region, effective size, and traceplots

# BEGIN SOLUTION
samples <- mh_logistic(10000, 10000, c(-1, 1))

ld50_ess_mh <- NULL # Save effective sample Size of ld50 here
```

Problem 2. Implementing your own Hamiltonian Monte Carlo

2a. Hamiltonian Monte Carlo can improve upon "vanilla" Metropolis-Hastings by leveraging information about the gradient of the posterior. Write out the gradient of the log posterior and fill in the function below. The function should return the gradient of the log posterior at any point in the parameter space.

```r
# Return the gradient as a vector of length 2
# first element is dl/dalpha and second is dl/dgamma
grad_log_posterior <- function(alpha, gamma){
```

```
  ## Fill in
}
```

We can use the following code, which computes a numeric approximation to the derivative, to test that we computed our derivative correctly. This numeric approximation is about 5x slower than the exact derivative (if interested, you can benchmark the run time of the functions yourself using `microbenchmark`).

```
numeric_grad_log_posterior <- function(alpha, gamma, eps){

  alpha_deriv <- (log_posterior(alpha+eps, gamma) - log_posterior(alpha-eps, gamma))/(2*eps)
  gamma_deriv <- (log_posterior(alpha, gamma + eps) - log_posterior(alpha, gamma - eps))/(2*e
  c(alpha_deriv, gamma_deriv)
}
```

Evaluate the cell below to test that your derivative is correct.

```
test_that("q1b", {
  expect_equal(grad_log_posterior(-10, 1),
               numeric_grad_log_posterior(-10, 1 , eps=1e-6), tol=1e-4)
})

test_that("q1b", {
  expect_equal(grad_log_posterior(-15, 2),
               numeric_grad_log_posterior(-15, 2, eps=1e-6), tol=1e-4)
})
```

2b. In the function skeleton provided, implement an HMC sampler. At each iteration, use $L = 10$ leapfrog steps with $\epsilon = 0.1$ as the suggested defaults in BDA. Set the mass matrix to the identity. Save the effective sample size of γ in a variable called `ld50_ess_hmc`. If you like, experiment with different mass matrices and find a mass matrix which maximizes the effective sample size.

```
#' HMC
#'
#' @param samples
#' @param burnin
#' @param log_post
#' @param grad_log_post
#' @param init
#' @param mass_mat
```

```r
#' @param L
#' @param eps
#'
#' @return
#' @export
#'
#' @examples
hmc <- function(samples=10000, burnin=samples, log_post, grad_log_post, init=c(1, 1), mass_ma

  accepts <- rep(0, samples+burnin)
  samps <- matrix(0, nrow=samples+burnin, ncol=2, dimnames=list(1:(burnin+samples), c("alpha"

  pars <- init

  for(iter in 1:(samples+burnin)){

    ## HMC Alg here

    samps[iter, ] <- NULL ## theta_t here
  }

  if(burnin > 0) {
    samps <- samps[-(1:burnin), ]
    accepts <- accepts[-(1:burnin)]
  }

  return(list(samps=samps, accepts=accepts))

}
```

```r
mass_mat <- diag(2)
L <- 10
eps <- .1
out <- hmc(samples=10000, burnin=1000,
           log_posterior, grad_log_posterior, init=c(1, 1),
           mass_mat=mass_mat, L=L, eps=eps)

ld50_ess_hmc <- NULL ## ESS for ld50 in HMC
```

Stan implementation

3a. Now implement your model in stan and run one chain with a warmup of 10000 and generate 10000 samples just as above. Save the effective sample size of γ in a variable called `ld50_ess_nuts`. Make a scatter plot with α samples on the x-axis and γ-samples on the y-axis (e.g using `bayesplot::mcmc_scatter`). Explain why the shape of the posterior dependence between α and γ makes sense.

```
ld50_ess_nuts <- NULL ## ESS from Stan
```

3b. Comparing the effective size from all iterations

The effective sample sizes for each implementation are:

Give at least two reasons why looking at effective sample size alone is not enough to tell us which algorithm is performed the best.

Visualizing uncertainty in the inferred regression function

In this part, we will use the MCMC samples from Stan to plot the posterior distribution of $\theta(x_i)$ as a function of x. We'll make the plot on a grid of values from 0.5 to 2.

The following code generates a dataframe where the hive death probabilities are computed for each of the 10000 samples. What is `map2` and `unnest` doing here?

```
xgrid <- seq(0.5, 2, by=0.1)

compute_prob <- function(alpha, gamma) {
  exp(alpha - alpha/gamma*xgrid) / (1 + exp(alpha - alpha/gamma*xgrid))
}

df <- stan_res %>%
  spread_draws(alpha, gamma) %>%
  mutate(x=list(xgrid),
         prob=map2(alpha, gamma,  compute_prob)) %>%
  unnest(c(x, prob))
```

Use geom_lineribbon to plot the posterior distribution of θ given x as a ribbon plot.