BDA - Assignment 5

Anonymous

22/3/2020

Bioassay with Metropolis

Replicating the Metropolis algorithm for the Bioassay dataset

Q1 - Implementing the Metropolis algorithm as an R function for the bioassay dataset

We use the following Gaussian prior:

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} \sim \mathcal{N}\{\mu_{\mathbf{0}}, \mathbf{\Sigma_0}\}$$

Where

$$\mu_{\mathbf{0}} = \begin{pmatrix}\mu_\alpha \\ \mu_\beta\end{pmatrix} = \begin{pmatrix}0 \\ 10\end{pmatrix}$$

and

$$\mathbf{\Sigma_0} = \begin{pmatrix}\sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2\end{pmatrix} = \begin{pmatrix}4 & 10 \\ 10 & 100\end{pmatrix}$$

.

a)

Implementing the density rate:

```
# From ex 4 we have our p_log_prior and p_log_posterior functions:

mu_alpha <- 0; sd_alpha <- 2
mu_beta <- 10; sd_beta <- 10
rho <- 0.5 # correlation

p_log_prior <- function(alpha, beta){
  x = cbind(alpha, beta)
  d <- dmvnorm(x, mean=c(mu_alpha, mu_beta),
               sigma = matrix(c(sd_alpha^2, rho*sd_alpha*sd_beta, rho*sd_alpha*sd_beta, sd_beta^2 ),
                              ncol=2))
  return(log(d))
}

p_log_posterior <- function(alpha, beta, x=bioassay$x, y=bioassay$y, n=bioassay$n){
  p_log_likelihood <- bioassaylp(alpha, beta, x, y, n)
```

```
  post <- p_log_prior(alpha, beta) + p_log_likelihood
  return(post)
}

# we will use these in our density_ratio:
density_ratio <- function(alpha_propose, alpha_previous,
                          beta_propose, beta_previous,
                          x = bioassay$x, y = bioassay$y, n = bioassay$n)
  {
  p1 <- p_log_posterior(alpha_propose, beta_propose, x=bioassay$x, y=bioassay$y, n=bioassay$n)
  p0 <- p_log_posterior(alpha_previous, beta_previous, x=bioassay$x, y=bioassay$y, n=bioassay$n)
  ratio <- exp(p1 - p0)
  return(ratio)
}

# test, result should be 0.84
density_ratio(alpha_propose = 0.374, alpha_previous = 1.89, beta_propose = 20.04, beta_previous = 24.76
              x = bioassay$x,
              y = bioassay$y,
              n = bioassay$n)
```

```
## [1] 0.8420882
```

b) Implementing the metropolis algorithm using denstity ratio:

Creating a function "metropolis_bioassay()" that implements the metropolis algorithm. We use a normal proposal distribution from the text:

$$\alpha^* \sim \mathcal{N}\{\alpha_{t-1}, \sigma = 1\}$$

and

$$\beta^* \sim \mathcal{N}\{\beta_{t-1}, \sigma = 5\}$$

.

```
# The proposal distribution:
proposal_distribution <- function(param){
  sigma = matrix(c(1, 2, 2, 5), ncol=2)
  return(rnorm(2, mean = param, sd = sigma))
}

# The Metropolis function:
metropolis_bioassay <- function(startvalue, iterations){
  chain = array(dim = c(iterations+1,2))
  chain[1,] = startvalue
  for (i in 1:iterations){
    proposal <- proposal_distribution(chain[i,])
    r <- density_ratio(alpha_propose = proposal[1], alpha_previous = chain[i,1],
                       beta_propose = proposal[2], beta_previous = chain[i,2],
                       x = bioassay$x, y = bioassay$y, n = bioassay$n)
    if (runif(1) < r){
      chain[i+1,] = proposal
    }else{
      chain[i+1,] = chain[i,]
```

```
    }
  }
  return(chain)
}
```

Q2 - Metropolis continued..

Including the number of chains used, the starting pints, tje number of draws and the warm-up length:

```
n_chains <- 5 # numbers of chain
iterations <- 50000 # number of draws
startvalues <- matrix(c(-1,1,-5,5,-2,2,4,-5,10,-1),
                      byrow = T, ncol=2)
warm_up <- iterations/2 #warmup length

run_in_chains <- function(n_chains, iterations, startvalues){
  m <- n_chains*2
  n <- round(iterations/2/2)
  chains <- array(dim = c(2*n,2, n_chains))
  for(j in 1:n_chains){
    chain <- metropolis_bioassay(startvalues[j,], iterations)
    warm_up <- iterations/2
    new_chain <- chain[-(1:warm_up),]
    new_chain <- new_chain[1:warm_up,]
    chains[, ,j] <- new_chain
  }
  return (chains)
}

chains <- run_in_chains(n_chains, iterations, startvalues)
```

Missing the plot...

Q3 - Convergence analysis

$\hat{R}$ for convergence analysis using the BDA3 version (11.4), where $\hat{R}$ compares the between- and within-chain estimates for model parameters

```
R_hat <- function(chains){
  m <- 2*dim(chains)[3]
  n <- round(dim(chains)[1]/2)
  psi_ij <- array(dim = c(n, 2, m))
  for(j in 1:n_chains){
    psi_ij[, , 2*j-1] <- chains[(1:n), , j]
    psi_ij[, , 2*j] <- chains[-(1:n), , j]
  }

  psi_j <- array(dim = c(m, 2))
  s_j_2 <- array(dim = c(m, 2))
  for(j in 1:m){
    psi_j[j,] <- colMeans(psi_ij[, ,j])
```

```
    s_j_2[j,] <- rowSums(apply(psi_ij[, ,j], 1, function(x) x-psi_j[j,])^2) / (n-1)
  }
  psi <- colMeans(psi_j)

  B <- ( n/(m-1) ) * sum(apply(psi_j, 1, function(x) x-psi)^2)
  W <- colMeans(s_j_2)
  var_hat <- ((n-1)/n) * W  + (1/n) * B
  R_hat <- sqrt(var_hat/W)
  return(R_hat)
}
```

Q4 - Plots

Scatter plotting the draws for α and β

```
samp_A <- chains[, 1, 1]
samp_B <- chains[, 2, 1]
xl <- c(-3, 8); yl <- c(-5, 35)
ggplot(data = data.frame(samp_A, samp_B)) +
  geom_point(aes(samp_A, samp_B), color = 'darkgreen', size = 0.2) +
  coord_cartesian(xlim = xl, ylim = yl) +
  labs(x = 'alpha', y = 'beta')
```