Introduction to Bayesian Statistics with R

3: Exercises

Jack Kuipers

13 May 2024

Exercise 3.1 - MCMC

For MCMC we can walk randomly and accept according the the MH ratio to eventually sample proportionally to any target distribution $p(x)$

```r
# simple MCMC function
# n_its is the number of iterations
# start_x the initial position
# rw_sd is the sd of the Gaussian random walk
basicMCMC <- function(n_its = 1e3, start_x = 0, rw_sd = 1, ...) {
  xs <- rep(NA, n_its) # to store all the sampled values
  x <- start_x # starting point
  xs[1] <- x # first value
  p_x <- target_density(x, ...) # probability density at current value of x
  for (ii in 2:n_its) { # MCMC iterations
    x_prop <- x + rnorm(1, mean = 0, sd = rw_sd) # Gaussian random walk to propose next x
    p_x_prop <- target_density(x_prop, ...) # probability density at proposed x
    if (runif(1) < p_x_prop/p_x) { # MH acceptance probability
      x <- x_prop # accept move
      p_x <- p_x_prop # update density
    }
    xs[ii] <- x # store current position, even when move rejected
  }
  return(xs)
}
```

For example, if we want to sample from a Student-t distribution we can use the following target

```r
target_density <- function(x, nu) {
  dt(x, nu) # Student-t density
}
```

and run a short chain with $\nu = 5$

```r
basicMCMC(nu = 5)
```

Examine the output MCMC chain for different lengths. How many samples would we need to get close to the Student-t distribution?

Use the samples to estimate (see also the description in Bonus Exercise 3.3)

$$\int \cos(t) f_5(t) \mathrm{d}t \,, \qquad f_\nu(t) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

where $f_\nu(t)$ is the probability density of a Student's t-distribution with ν degrees of freedom.

Bonus Exercise 3.2 - HMC

NOTE: This exercise is an optional bonus for when you have sufficient free time.

In HMC we move in the constant energy space (using Hamiltonian mechanics) after we add another Gaussian dimension, rather than randomly. With perfect propagation we would always accept the move, allowing us to explore the space more efficiently. For computational efficiency, we prefer to propagate numerically with a faster and more approximate method (normally the leapfrog) and then accept according the the MH ratio to eventually sample proportionally to any target distribution $p(x)$.

We define $U(x) = -\log(p(x))$, and with a standard normal for the extra dimension ρ, we have the Hamiltonian

$$H(x, \rho) = U(x) + \frac{\rho^2}{2}$$

while Hamilton's equations give us the dynamics:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial H}{\partial \rho} = \rho \,, \quad \frac{\mathrm{d}\rho}{\mathrm{d}t} = -\frac{\partial H}{\partial x} = -\frac{\mathrm{d}U}{\mathrm{d}x}$$

To propagate under these dynamics, we also need the gradient of the target, so let's update our definition above for the Student-t while keeping it back-compatible with before

```r
target_density <- function(x, nu, grad = FALSE) {
  dens <- dt(x, nu) # Student-t density
  if (grad) { # return density and gradient
    grad <- -(nu + 1)/(nu + x^2)*x*dens
    return(list(dens = dens, grad = grad))
  } else { # return just the density
    return(dens)
  }
}
```

and define our function U

```r
U_fn <- function(x, ...) {
  p_x <- target_density(x, ..., grad = TRUE)
  U <- -log(p_x$dens)
  grad <- -1/p_x$dens*p_x$grad
  return(list(U = U, grad = grad))
}
```

Now we're ready for our HMC code. This is similar to the MCMC code, with an internal loop for the leapfrog propagation. Since this takes L steps, we shorten the number of outside iterations a bit to compensate.

```r
# simple HMC function
# n_its is the number of iterations
# start_x the initial position
# L is the number of steps of numerical propagation
# under the Hamiltionian H = U + rho^2/2, U = -log(target_density)
# epsilon is the size of the steps
```

```r
basicHMC <- function(n_its = 1e2, start_x = 0, L = 10, epsilon = 0.1, ...) {
  xs <- rep(NA, n_its) # to store all the sampled values
  x <- start_x # starting point
  xs[1] <- x # first value
  U_x <- U_fn(x, ...) # log density and gradient at current x
  for (ii in 2:n_its) { # HMC iterations
    rho <- rnorm(1) # normal sample (we could define scheme with different sd)
    x_prop <- x
    # Leapfrog method to propagate under Hamiltonian:
    rho_prop <- rho - epsilon/2*U_x$grad # half step for momentum
    for (j in 1:L) {
      x_prop <- x_prop + epsilon*rho_prop # position update
      U_prop <- U_fn(x_prop, ...) # update gradient
      # update momentum, with a half step at the end
      rho_prop <- rho_prop - epsilon*U_prop$grad/(1 + (j==L))
    }
    MH_prob <- exp(U_x$U + rho^2/2 - U_prop$U - rho_prop^2/2)
    if (runif(1) < MH_prob) { # MH acceptance probability
      x <- x_prop # accept move
      U_x <- U_prop # update density
    }
    xs[ii] <- x # store current position, even when move rejected
  }
  return(xs)
}
```

Now we can run a short chain again with $\nu = 5$

```r
basicHMC(nu = 5)
```

and examine the output HMC chain for different lengths. How many samples do we now need to get close to the Student-t distribution?

Do we get good estimates for the integral from before?

Bonus Exercise 3.3 - Monte Carlo integration

NOTE: This exercise is an optional bonus for when you have sufficient free time.

Computing expectations can be applied to any continuous function

$$E[g(x)] = \int g(x)p(x)\mathrm{d}x$$

so that integrals where we recognise $p(x)$ as (proportional to) a probability distribution may be estimated with Monte Carlo methods since

$$E[g(x)] \approx \frac{1}{M}\sum_{i=1}^{M} g(x_i)$$

for M random samples x_i sampled according to $p(x)$. Use samples from a Gaussian to estimate the following three integrals:

$$\int |x|\mathrm{e}^{-x^2}\mathrm{d}x\,, \qquad \int \sin(x)\mathrm{e}^{-x^2}\mathrm{d}x\,, \qquad \int \cos(x)\mathrm{e}^{-x^2}\mathrm{d}x$$

Reminder, the Gaussian probability density has the following general form:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Compare the estimated values to the exact values of the integrals.

NOTE: For the comparison to the real values, you can integrate analytically or use `R`'s `integrate` function. The errors from the true value, like standard errors in general, decrease like $\frac{1}{\sqrt{M}}$.