MTH 511a - 2020: Lecture 7

Instructor: Dootika Vats

1 Generating continuous random variables

1.1 Accept-reject method: intuition

Algorithm 1 Accept-reject for continuous random variables

1: Draw $U \sim U[0,1]$

2: Draw proposal $Y \sim G$

3: **if** $U \leq \dfrac{f(Y)}{c\, g(Y)}$ **then**

4: Return $X = Y$

5: **else**

6: Go to Step 1.

At a proposed value y:

- if $f(y)$ is large but $g(y)$ is small means this value will not be proposed often and is a good value to accept for f, so higher probability of accepting it.
- if $f(y)$ is small but $g(y)$ is large, then this value will be proposed often but is unlikely for f, so accept this value less often.

We can choose any g we want as long its support is larger than other the support of f. However, some gs will be better than other gs.

1.2 Accept-reject method: examples

Example 1. **Beta distribution:** Consider the beta distribution Beta$(4, 3)$, where

$$f(x) = \frac{\Gamma(7)}{\Gamma(4)\Gamma(3)} x^{4-1}(1-x)^{3-1} \quad 0 < x < 1; \quad .$$

Consider a uniform proposal distribution. So that $G = U(0, 1)$ and

$$g(x) = 1 \qquad \text{for } x \in (0, 1)\,.$$

For this choice of g,

$$\sup_{x\in(0,1)} \frac{f(x)}{g(x)} = \sup_{x\in(0,1)} f(x)$$

We can show that maximum of $f(x)$ occurs at $x = 3/5$ and

$$\sup_{x\in(0,1)} \frac{f(x)}{g(x)} = \sup_{x\in(0,1)} f(x) = 60\left(\frac{3}{5}\right)^3\left(\frac{2}{5}\right)^2 = c\,.$$

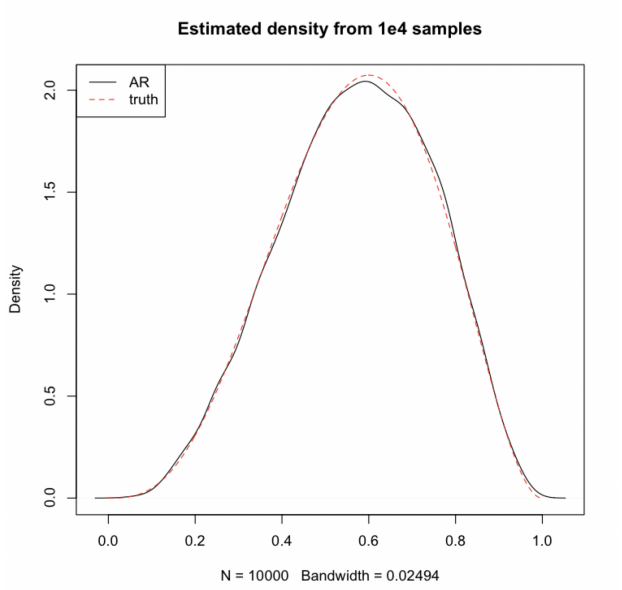
Algorithm 2 Accept-reject for Beta$(4, 3)$

1: Draw $U \sim U[0, 1]$
2: Draw proposal $Y \sim U(0, 1)$
3: **if** $U \le \dfrac{f(Y)}{c\, g(Y)}$ **then**
4: Return $X = Y$
5: **else**
6: Go to Step 1.

```
#########################
### Accept-reject for
## Beta(4,3) distribution
#########################
set.seed(1)
beta_ar <- function()
{
  c <- 60 *(3/5)^3 * (2/5)^2
  accept <- 0
  counter <- 0  # count the number of loop
  while(accept == 0)
  {
    counter <- counter + 1
```

```
      U <- runif(1)
      prop <- runif(1)

      ratio <- dbeta(prop, shape1 = 4, shape2 = 3)/c

      if(U <= ratio)
      {
         accept <- 1
         return(c(prop, counter))
      }
   }
}

N <- 1e4
samp <- numeric(length = N)
counts <- numeric(length = N)
for(i in 1:N)
{
   foo <- beta_ar() # I use foo as a dummy name
   samp[i] <- foo[1]
   counts[i] <- foo[2]
}
```

```
x <- seq(0, 1, length = 500)
plot(density(samp), main = "Estimated density from 1e4 samples")
lines(x, dbeta(x, shape1 = 4, shape2 = 3), col = "red", lty = 2)
legend("topleft", lty = 1:2, col = c("black", "red"), legend = c("AR",
    "truth"))
```



```r
# This is c
(c <- 60 *(3/5)^3 * (2/5)^2)
# [1] 2.0736

# This is the mean number of loops required
mean(counts)
# [1] 2.0776

#They are almost the same!
```

Example 2. **Normal distribution**

What would be a good proposal distribution for $N(0,1)$. It is always good to consider distributions that have "fatter tails" than our target distribution. These may lead to more rejections, but at least we will propose values from the tails.

The target density function is

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,.$$

We know that the t-distribution has the right support and fatter tails and the "fattest" t distribution is Cauchy. The pdf of a Cauchy distribution is

$$g(x) = \frac{1}{\pi}\frac{1}{1+x^2}\,.$$

We will need to find the supremum of the ratio of the densities. Consider

$$\frac{f(x)}{g(x)} = \frac{\pi}{\sqrt{2\pi}}(1+x^2)e^{-x^2/2}\,.$$

The supremum above occurs at $x = -1, 1$, so

$$\sup_{x\in\mathbb{R}} \frac{f(x)}{g(x)} = \frac{f(1)}{g(1)} = \sqrt{2\pi}e^{-1/2} \approx 1.746 \Rightarrow c = 1.80\,.$$

You can implement the algorithm similarly.

Example 3. **Sampling from a uniform circle** Consider a unit circle centered at $(0,0)$:

$$x^2 + y^2 < 1\,.$$

We are interested in sampling uniformly from within this circle. The target density is

$$f(x,y) = \frac{1}{\pi} I(x^2+y^2<1)\,.$$

Consider the uniform distribution on the square as a proposal distribution

$$g(x,y) = \frac{1}{4} I(-1 < x < 1) I(-1 < y < 1)\,.$$

First, we will find c. Consider

$$\frac{f(x,y)}{g(x,y)} = \frac{4}{\pi} I(x^2 + y^2 < 1) \leq \frac{4}{\pi} := c\,.$$

Next, note that

$$\frac{f(x,y)}{cg(x,y)} = \frac{4}{\pi} I(x^2 + y^2 < 1) \frac{\pi}{4} = I(x^2 + y^2 < 1)\,.$$

So for any (x,y) drawn from within the square, the ratio will be either 1 or 0, thus, no need to draw a uniform at all!

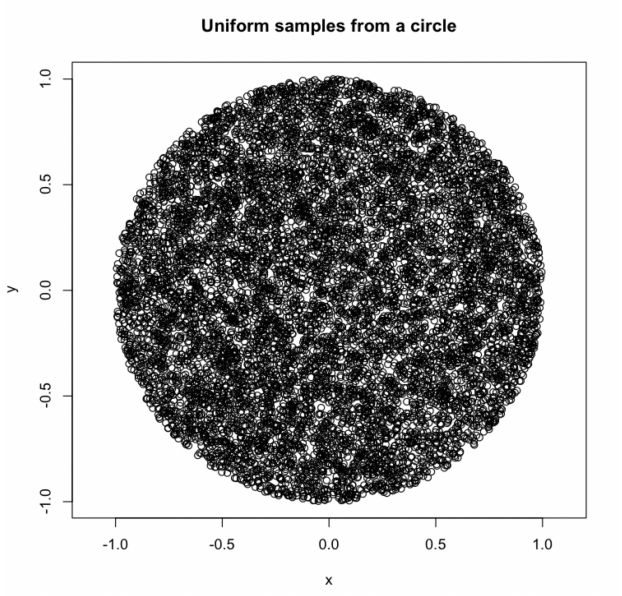
Algorithm 3 Accept-reject for Uniform distribution on a circle

1: Draw proposal $(U_1, U_2) \sim U(-1,1) \times U(-1,1)$
2: **if** $U_1^2 + U_2^2 \leq 1$ **then**
3: Return $(X,Y) = (U_1, U_2)$
4: **else**
5: Go to Step 1.

```
set.seed(1)
circle_ar <- function()
{
   accept <- 0
   counter <- 0  # count the number of loop
   while(accept == 0)
   {
      counter <- counter + 1
      prop <- runif(2, min = -1, max = 1) #c(U1, U2)

      in.or.out <- prop[1]^2 + prop[2]^2 < 1

      if(in.or.out)
      {
         accept <- 1
         return(c(prop, counter))
      }
   }
}

N <- 1e4
```

```r
samp <- matrix(0, ncol = 2, nrow = N)
counts <- numeric(length = N)
for(i in 1:N)
{
   foo <- circle_ar() # I use foo as a dummy name
   samp[i,] <- foo[1:2]
   counts[i] <- foo[3]
}
```

```r
plot(samp[,1], samp[,2], xlab = "x", ylab = "y",main = "Uniform samples
    from a circle", asp = 1)
```




```r
4/pi
# [1] 1.27324
mean(counts) # very close
# [1] 1.2783
```

1.3 Questions to think about

- How can you decide whether one proposal distribution is better than another proposal distribution?
- Try implementing the circle/square example in 3 dimension, 4 dimensions, and a general p dimensions. How happens to c?
- Can a similar A-R algorithm be implemented for Beta(m, n) for all $m, n \in \mathbb{Z}$?