MTH 511a - 2020: Lecture 4

Instructor: Dootika Vats

1 The Acceptance-Rejection Technique

Although we can draw from any discrete distribution using the inverse transform method, you can imagine that for distributions on countably infinite spaces (like the Poisson distribution), the inverse transform method may be very expensive. In such situations, acceptance-rejection sampling may be more reliable.

Let $\{p_j\}$ denote the pmf of the target distribution with $\Pr(X = a_j) = p_j$ and let $\{q_j\}$ denote the pmf of another distribution with $\Pr(Y = a_j) = q_j$. Suppose you can efficiently draw from $\{q_j\}$ and you want draw from $\{p_j\}$. Let c be a constant such that

$$\frac{p_j}{q_j} \leq c \quad \text{for all } j \text{ such that } p_j > 0\,.$$

If we can find such a $\{q_j\}$ and c, then we can implement an *Acceptance-Rejection* or *Accept-Reject* sampler. The idea is to draw samples from $\{q_j\}$ and accept these samples if they seem likely to be from $\{p_j\}$.

Algorithm 1 Acceptance-Rejection sampler to draw 1 sample from $\{p_j\}$

```
1: Draw U ~ U[0,1]
2: Simulate Y = y with probability mass function q_j
3: if U < p_y / (c q_y) then
4:     Return X = y and stop
5: else
6:     Goto step 1
```

Theorem 1. *The Accept-Reject method generates a random variable with probability*

$$\Pr(X = a_j) = p_j \,.$$

Further, the number of iterations needed to generate an acceptance is distributed as Geometric$(1/c)$.

Proof. First, we look at the second statement. We note that the number of iterations required to stop the algorithm is clearly geometrically distributed by the definition of the geometric distribution.

We will show that the probability of success is $1/c$. "Success" here is an acceptance. First, consider

$$\begin{aligned}\Pr(Y = a_j, \text{accepted}) &= \Pr(Y = a_j)\ \Pr(\text{Accept} \mid Y = a_j) \\ &= q_j \Pr\left(U < \frac{p_j}{cq_j}\right) \\ &= q_j \frac{p_j}{cq_j} = \frac{p_j}{c} \,.\end{aligned}$$

Using this we can calculate the marginal distribution of accepting is

$$\Pr(\text{accept}) = \sum_j \Pr(Y = a_j, \text{accept}) = \sum_j \frac{p_j}{c} = \frac{1}{c} \,.$$

Thus, the second statement is proved. We will now use this to show that the

We move on to the first statement. Note that

$$\begin{aligned}\Pr(X = a_j) &= \sum_{n=1}^{\infty} \Pr(a_j \text{ accepted on iteration } n) \\ &= \sum_{n=1}^{\infty} \Pr(\text{No acceptance until iteration } n-1)\ \Pr\left(Y = a_j, \text{accept}\right) \\ &= \sum_{n=1}^{\infty} \left(1 - \frac{1}{c}\right)^{n-1} \frac{p_j}{c} \\ &= p_j \,.\end{aligned}$$

□

One important thing to note is that within the support $\{a_j\}$ of $\{p_j\}$, the proposal distribution must always be positive. That is, for all a_j, $\Pr(Y = a_j) = q_j > 0$. In other words, a proposal distribution must have support *larger* than the target distribution.

Example 1 (Sampling from Binomial using AR). The binomial distribution has pmf

$$\Pr(X = x) = \binom{n}{x} (1-p)^{n-x} p^x \quad \text{for } x = 0, 1, \ldots, n \,.$$

We will use AR to simulate draws from Binomial(n, p).

We could use any of Poisson, negative-binomial, or geometric distributions. We choose to use the geometric distribution, but we must be a little careful.

We use the version of geometric distribution that is defined at the number of failures before the first success, so that the support of the geometric distribution has 0 in it. The pmf of the geometric distribution is

$$\Pr(X = x) = (1-p)^x p \quad x = 0, 1, \dots .$$

We will first find c. Note that

$$\begin{aligned} \frac{p(x)}{q(x)} &= \frac{\binom{n}{x}(1-p)^{n-x}p^x}{(1-p)^x p} \\ &= \binom{n}{x}(1-p)^{n-2x}p^{x-1} . \end{aligned}$$

Set

$$c = \max_{x=0,1,\dots,n} \binom{n}{x}(1-p)^{n-2x}p^{x-1} .$$

For $n = 10, p = 0.25$, we yield $c = 2.373\dots$.

To be safe (since I don't know all the decimal points), we can set $c = 2.5$. Now the AR algorithm can be implemented simply. Below is code for the Accept-Reject sampler.

```
############################################
## Accept Reject algorithm to draw from
## Binomial(n,p)
############################################
set.seed(1)

# Function draws one value from Binom(n,p)
# n = number of trials
# p = probability of success
draw_binom <- function(n, p)
{
  accept <- 0

  # upper bound calculated in the notes
  x <- 0:n
  all_c <- choose(n,x) * (1-p)^(n - 2*x) * p^(x-1)
  c <- max(all_c) + .001 # final c with slight increase for numerical
     stability.

  while(accept == 0)
  {
   U <- runif(1)
```

```
    prop <- rgeom(1, prob = p) #draw proposal

    ratio <- dbinom(x = prop, size = n, prob = p)/
      (c* dgeom(x = prop, prob = p))
    if(U < ratio)
    {
      accept <- 1
      rtn <- prop
    }
  }
  return(rtn)
}
draw_binom(n = 10, p = .25)
# [1] 4

###
# If we want X1, ..., Xn ~ Binom(n.p)
# we need to call the function multiple times

N <- 1e3 # sample size
samp <- numeric(N)
for(t in 1:N)
{
  samp[t] <- draw_binom(n = 10, p = .25)
}
mean(samp) #should be n*p = 2.5
# [1] 2.51
```

1.0.1 Question to think about

- Why is c always greater than 1?
- Can we always find such a c?
- What happens when c is large or small?