Tutorial: Rejection Sampling

Song Liu (song.liu@bristol.ac.uk)
GA 18, Fry Building,
Microsoft Teams (search "song liu").

Sampling

- Sampling is one of the most common tasks in statistical data analysis and in particular Monte Carlo simulations.
- The task of sampling is to draw samples of a random variable, given a probability density/mass function.

Sampling

- For example, how do you generate samples from a normal distribution $N(\mu, \sigma)$ given its probability density function $p(\boldsymbol{x}) \propto \exp(-(x-\mu)^2/2\sigma^2)$?
- Of course, most data science programming languages have dedicated bulit-in function to generate samples from normal distributions.
  - ```
    a <- rnorm(1000, mean = 1, std = .5)
    ```
  - Many basic IoT devices does not have full support of a wide variety of statistical functions.
    ```

Rejection Sampling

- Suppose we have a uniform sampler that samples observations from $U(0,1)$ and a basic sampler for a known distribution with density function $q(x)$
- Target: sample $n_{\max}$ observations from a distribution with density function $p(x) \propto \bar{p}(x)$.
- Algorithm:
  - Draw a sample x from $q(x)$ and a uniform sample $u \sim U(0,1)$
  - If $u < \frac{p(x)}{Mq(x)}$, accept sample x.
    - If # accepted samples equals to $n_{\max}$, quit.
  - Repeat the algorithm.

Rejection Sampling

- M in the previous algorithm is a big number, which needs to satisfy $\frac{p(x)}{q(x)} \leq M$.
    - In our toy experiment, we can set M to be a big number, say 100 or 1000.
- When ``accepting'' a sample, we save it to a vector, so the vector later contains only accepted samples.

Vector Appending

- If `a` is a length-k vector, `a[k + 1] = C` will append another element `C` at the end of `a`.

```
a <- c(1,2,3,4)
a[5] <- 5
print(a)
[1] 1 2 3 4 5
```

Code Skeleton

```
# how many samples do we want?
n_max = 1000

pbar <- function(x){
  # the PDF of distribution, from which you want to sample.
  return(exp(-(x-1)^2/.5))
}

acc <- c() # create an empty vector
n <- 0 # how many samples have we already obtained?
M <- 200 # M = 200
while(n < n_max){
  u <- runif(1) # generate a uniform sample
  x <- rnorm(1) # in this example, q(x) is N(0,1)

  # TODO: complete the algorithm here

}

print(mean(acc))
print(sd(acc))
```

Questions:

1. Complete the above code skeleton according to the rejection sampling algorithm.
2. What output is expected when the algorithm is implemented correctly?
3. Fill out the blank: The bigger M is, the ___ (faster/slower) the algorithm becomes.