AIM101-Tutorial-2025-04-07

Raghavan

April 2025

1. (**Theory**) Show using the inverse CDF method that to generate samples from the geometric distribution $g(s) = p(1-p)^s$ (probability of success happening exactly at the $(s+1)$th iteration, p being the success probability in any isolated iteration) for a fixed parameter $p \in [0,1]$, we can generate a uniform random sample $u \sim [0,1]$ and then get S as $S = \left\lfloor \frac{lnu}{\ln(1-p)} \right\rfloor$.

2. (**Implementation**) The above exercise has applications to sampling from a very large file on the disk — we need a non-repeating, order-preserving uniform sample from the records in this file, but we have only sequential access (as is typically the case for files in external memory) and the size is too big for us to keep a explicit table of indexes for the records in the RAM. It turns out that we can do it using the following algorithm (read p in step 7 as f_p) where $S(n,N)$ is a random variable with a very com-

    1: Initialize Records to be processed $N \leftarrow N_f$
    2: Initialize Records Required for the sample $n \leftarrow n_s$
    3: Initialize Current File Pointer $f_p \leftarrow 0$
    4: **while** f_p not end of file **do**
    5: Generate a random $s \leftarrow S(n,N)$.
    6: Skip the next s records — $f_p \leftarrow f_p + s$.
    7: Select the record at p as the next sample — $n \leftarrow n-1$, $N \leftarrow N-s-1$
    8: $f_p \leftarrow f_p + 1$.
    9: **end while**

    plex distribution having the density function $f(s) = \frac{n}{N} \frac{\prod_{i=1}^{s}(N-n-i+1)}{\prod_{i=1}^{s}(N-i)}$. (we will not prove this here — if you are interested refer to this paper *Jeffrey Scott Vitter. "Faster Methods for Random Sampling". In: Communications of the ACM 27.7 (July 1984), pp. 703–718*). Therefore to accomplish step 5, we will use rejection sampling with $g(s)$ as in the earlier exercise as the distribution we will sample from.

    (a) (**Theory**) Show that for $p = \frac{n-1}{N-1}$ and $c = \frac{n}{n-1}\frac{N-1}{N}$, $\forall s : f(s) \leq c.g(s)$. This satisfies the conditions required for rejection sampling.

    Replace Step 5 in algorithm above with rejection sampling and implement the full algorithm. Verify empirically for some reasonably large files that

this algorithm indeed gives us uniformly random non-repeating, ordered samples from the file. The full algorithm is shown below.

1: Initialize Records to be processed $N \leftarrow N_f$
2: Initialize Records Required for the sample $n \leftarrow n_s$
3: Initialize Current File Pointer $f_p \leftarrow 0$
4: **while** f_p not end of file **do**
5: Compute $p \leftarrow \frac{n-1}{N-1}$, $c \leftarrow \frac{n}{n-1}.\frac{N-1}{N}$
6: SampleFound $\leftarrow$ False
7: **while** not SampleFound **do**
8: $u_1 \leftarrow$ uniformly distributed from the range $[0, 1]$
9: $s \leftarrow \left\lfloor \frac{\ln u_1}{\ln(1-p)} \right\rfloor$
10: $u_2 \leftarrow$ another uniformly distributed variable from the range $[0, 1]$
11: **if** $u_2 \leq \frac{f(s)}{c.g(s)}$ **then**
12: SampleFound $\leftarrow$ True
13: **end if**
14: **end while**
15: Skip the next s records — $f_p \leftarrow f_p + s$.
16: Select the record at f_p as the next sample — output this record
17: Update $n \leftarrow n - 1$, $N \leftarrow N - s - 1$, $f_p \leftarrow f_p + 1$.
18: **end while**

Use the following code template to read sequentially from a file

```
with open("filename") as fileobject:
    for line in fileobject:
        do_something_with(line)
```

3. (**Theory**) For the binomial distribution with parameter p, we know that the MLE based on m heads out of n tosses is $\hat{p} = (m/n)$ — this was done in the class. Show that this is an unbiased estimator (that $E[\hat{p}] = p$). Show that the MSE for this estimator is $p(1-p)/n$. Also take a different estimator $\tilde{p} = \frac{m+1}{p+2}$ that adds an artificial success and failure to the actual data. Show that $E[\tilde{p}] = \frac{np(1-p)}{(n+2)^2}$ and $MSE[\tilde{p}] = \frac{np(1-p)+(1-2p)^2}{(n+2)^2}$.

4. (**Implementation**) Plot MSE's of $\hat{p}$ and $\tilde{p}$ as functions of p and n and examine when $\tilde{p}$ may be preferable to $\hat{p}$.