WRITING A REJECTION SAMPLER

FRANK WOOD, BROOKS PAIGE

Introduction

In this exercise we will write a rejection sampler which draws from a Poisson distribution. An algorithm to sample a Poisson random variate k with rate λ is given by [1]. Initialize $L \leftarrow e^{-\lambda}$, $k \leftarrow 0$, and $p \leftarrow 1$, and then loop:

(1) Update $k \leftarrow k + 1$

(2) Sample $u \sim \text{Uniform}(0, 1)$

(3) Update $p \leftarrow p \times u$

(4) If $p \leq L$ return $k - 1$; otherwise repeat.

This algorithm increments k until eventually returning.

Note that each run of the program generates an execution trace with a possibly different amount of randomness; that is, sampling a value of $k = 10$ requires more random choices than a value of $k = 5$. If $\lambda = 4$, then under a Poisson distribution the probability masses $p(k = 3) = p(k = 4)$, but in this algorithm generating a $k = 4$ variate requires more random choices.

Particularly vexing, each random choice is a draw from $\text{Uniform}(0, 1)$, meaning that if we label all the random choices $u_1, \ldots, u_k$, then for each u_i we have $p(u_i) = 1$.

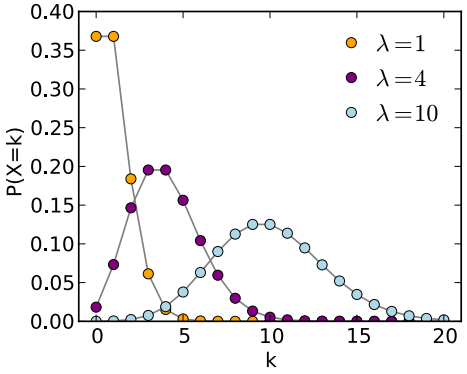

FIGURE 1. The probability mass function for a Poisson distributed random variable. Figure from Wikipedia: `http://en.wikipedia.org/wiki/Poisson_distribution`.

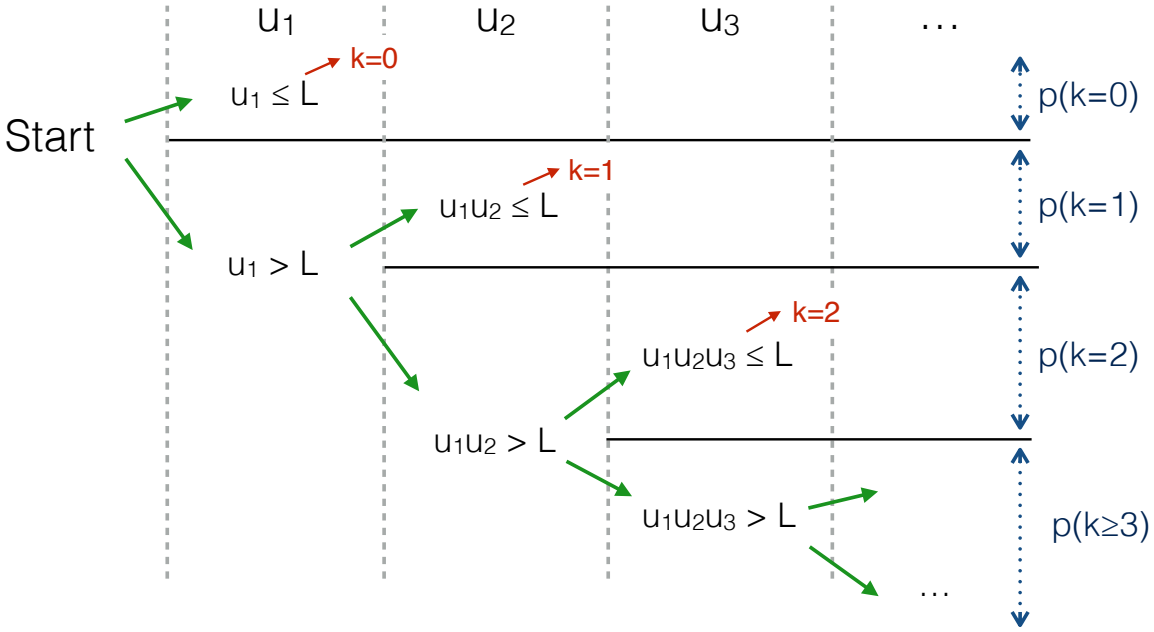

FIGURE 2. Diagram showing sequential steps during the execution of the sampler program. Execution flows to the right; each column shows a random draw $u_i \sim \text{Uniform}(0, 1)$. Green arrows represent the program branching that occurs at the end of each algorithm iteration. Red arrows denote program termination, with a given return value of k.

The execution path of this program is shown in Figure 2. Since the probability of the program execution trace is defined as the product of the probabilities of all random choices made in the execution of the program [3, 2], then each execution of the program has the same probability, regardless of k, with

$$p(\text{trace}) = \prod_{i=1}^{k} p(u_i) = 1 \tag{1}$$

for all $k = 0, 1, 2, \ldots$. The probability of sampling a particular value k is directly related to the length of the program execution trace — the algorithm is defined in such a way that, although each individual execution trace has equal probability, the number of execution traces which terminate after k total uniform draws is proportional to a Poisson distribution with rate λ.

We can compute the probability of this program halting (i.e. returning a value) after any particular number of random choices n have been made. Letting $L \equiv e^{-\lambda}$, we can compute these probabilities, for example

$$p(\text{halt} = 1) = p(u_1 \leq L) = L = e^{-\lambda} \tag{2}$$

which is exactly the probability that $k = 0$ under a Poisson distribution. We can compute this for higher numbers of random variables as well, although it becomes cumbersome:

$$
\begin{aligned}
p(\text{halt} = 2) &= p(u_1 u_2 \leq L, u_1 > L) && (3)\\
&= p(u_2 \leq \frac{L}{u_1}, u_1 > L) && (4)\\
&= \int_{u_1=L}^{1} \frac{L}{u_1} du_1 && (5)\\
&= -L \log L = \lambda e^{-\lambda}, && (6)
\end{aligned}
$$

which matches the Poisson mass $p(k = 1|\lambda)$, and

$$
\begin{aligned}
p(\text{halt} = 3) &= p(u_1 u_2 u_3 \leq L, u_1 > L, u_2 > \frac{L}{u_1}) && (7)\\
&= p(u_3 \leq \frac{L}{u_1 u_2}, u_1 > L, u_2 > \frac{L}{u_1}) && (8)\\
&= \int_{u_1=L}^{1} \int_{u_2=L/u_1}^{1} \frac{L}{u_1 u_2} du_2 du_1 && (9)\\
&= L \int_{u_1=L}^{1} \frac{1}{u_1} \int_{u_2=L/u_1}^{1} \frac{1}{u_2} du_2 du_1 && (10)\\
&= L \int_{u_1=L}^{1} \frac{-\log(L/u_1)}{u_1} du_1 && (11)\\
&= L \int_{u_1=L}^{1} \frac{\log(u_1) - \log(L)}{u_1} du_1 && (12)\\
&= L \left[\int_{u_1=L}^{1} \frac{\log(u_1)}{u_1} du_1 - \log(L) \int_{u_1=L}^{1} \frac{1}{u_1} du_1 \right] && (13)\\
&= L \left[\frac{-(\log L)^2}{2} + (\log L)^2 \right] && (14)\\
&= L \frac{(\log L)^2}{2} = e^{-\lambda} \frac{\lambda^2}{2} && (15)
\end{aligned}
$$

which matches the Poisson mass $p(k = 2|\lambda)$, and so on.

Questions:

(1) Write this sampler by filling in the missing function marked `<···>` in the following program:

```
[assume sample-poisson (lambda (rate) (begin
    (define L (exp (* -1 rate)))
    (define inner-loop (lambda (k p)
        <···>
    ))
    (inner-loop 1 (uniform-continuous 0 1))))]

[predict (sample-poisson 4)]
```

Confirm this function actually generates Poisson random variates.

(2) Review the two sampling approaches — particle Gibbs and random database — described in sections 3.1 and 3.2 of [3][1]. Although no posterior inference is being performed in this example, the two methods differ considerably in the manner by which consecutive samples are drawn. It is important to realize that for either approach, executing the program generates a *dependent* chain of Poisson random variates.

In random db, an initial sample is drawn by running the program forward once. Subsequent samples are proposed by

(1) selecting an individual random choice u_i from the uniform draws $u_1, \ldots, u_{k+1}$ instantiated in the current sample;

(2) resampling a program suffix beginning from u_i, by drawing a new value u_i', forgetting all random choices $u_{i+1}, \ldots, u_{k+1}$, and continuing program execution from the point at which u_i is drawn; this will create a new sequence of random choices $u_i', \ldots, u_{k'+1}'$ where k and k' are not necessarily the same;

(3) accepting or rejecting the new proposed set of random choices with probability given by Equation 5 in [3].

Assuming running `sample-poisson` generates an independent sample from a Poisson distribution, show that the random db algorithm over this program defines a Markov chain whose stationary distribution is the same Poisson.

References

[1] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, 1969.

[2] Brooks Paige and Frank Wood. A compilation target for probabilistic programming languages. In *ICML*, 2014.

[1] http://jmlr.org/proceedings/papers/v33/wood14.pdf

[3] F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, 2014.