Generating a random variable by coin tossing

Pablo Rotondo

March 4, 2017

1 Coin tossing

Suppose we are given a sequence of independent fair bits $X_1, X_2, X_3, \ldots$ (meaning X_i take 0 and 1 with probabily $1/2$) we want to produce with them a discrete random variable Y that takes the values $\{1, \ldots, k\}$ with probabilities $p_1, \ldots, p_k > 0$. The objective, of course, is to do this using the least possible number of bits.

Let us note that if we consider $X = 0.X_1X_2X_3\ldots$ in binary, this variable is uniformly distributed in $[0, 1]$. The usual way to produce Y given a number X uniform in $[0, 1]$ is to consider the intervals $I_1 = [0, p_1), I_2 = [p_1, p_1 + p_2), \ldots, I_n = [p_1 + \ldots + p_{n-1}, 1)$ and declare $Y = j$ when $X \in I_j$. Of course, knowing whether $X \in I_j$ or not, takes a few bits from $X_1, X_2, \ldots$. Indeed, having seen k digits, the only thing we know is that $X \in [X_1 \ldots X_k, X_1 \ldots X_k + 2^{-k}]$.

1.1 Uniform distribution $U_N \sim (1/N, \ldots, 1/N)$

The case of the uniform distribution is particularly interesting; it is indeed one of the most common distributions one might come across, but we claim that its study shows some interesting patterns too.

Algorithm 1. We begin with a simplified version that is slightly wasteful.

The idea is that we stop only when $[0.x_1 \ldots x_k, 0.x_1 \ldots x_k + 2^{-k}]$ is completely contained in one of the intervals $[0, 1/N), [1/N, 2/N), \ldots, [1 - 1/N, 1]$.

Why is this wasteful? Well, the probability of the resulting $x = 0.x_1x_2\ldots$ being a dyadic number $\frac{A}{2^B}$ is 0, so we could directly stop when

$$(0.x_1 \ldots x_k, 0.x_1 \ldots x_k + 2^{-k}) \subset I_m\,,$$

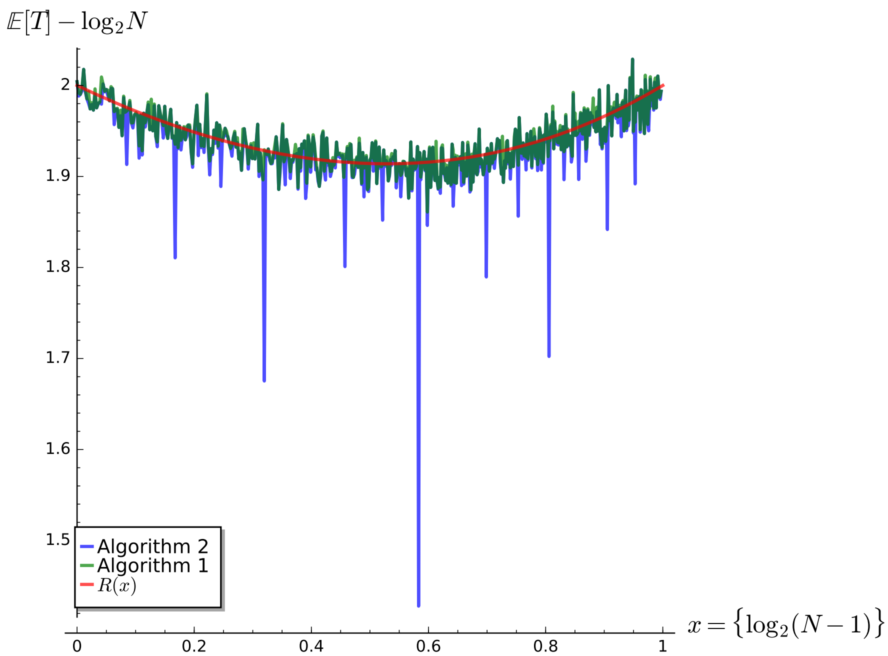

Figure 1: Experimental redundancy for $j = 9$. Algorithm 1 is the "wasteful algorithm", while $R(x)$ is the limit period for the redundancy.

thus neglecting the borders. This obviously improves the performance in cases like $N = 2^k$, in which we end up stopping after exactly k steps. The analysis of the algorithm also gets more interesting as we shall see afterwards.

See the corresponding code for our first (and wasteful) algorithm in Figure 2.

To start let us recall that

$$\mathbb{E}[T] = \sum_{k \geq 0} \mathbb{P}(T > k)\,,$$

which simplifies the computation of $\mathbb{E}[T]$, because telling whether we have to continue is easy: if there is a number of the form m/N in $[0.X_1 \dots X_k, 0.X_1 \dots X_k + 1/2^k]$, then we must conclude that $T > k$. This is so because m/N constitutes the border between two intervals $I_{m-1} = \left(\frac{m-1}{N}, \frac{m}{N}\right]$ and $I_m = \left(\frac{m}{N}, \frac{m+1}{N}\right]$ whenever $m \in \{1, \dots, N-1\}$.

For $2^{-k} > 1/N$, it is obviously true that $[0.x_1 \dots x_k, 0.x_1 \dots x_k + 1/2^k]$ contains a number of the form m/N with $m < N$, that is not $0.x_1 \dots x_k$. Indeed, notice that $0.x_1 \dots x_k < 1 - 1/N$.

If $N = 2^k$ then we observe that we may only stop if $[0.x_1 \dots x_k, 0.x_1 \dots x_k + 1/2^k]$ is $[1 - 2^{-k}, 1]$, hence in this case $\mathbb{P}(T > k) = \frac{N-1}{2^k}$.

Assume now that $2^{-k} < 1/N$, then there is exactly one number of the form $0.x_1 \dots x_k$ in the interval $\left[\frac{m}{N} - 2^{-k}, \frac{m}{N}\right)$, which is given by

```
x = 0
l = 0
while (true) :
  x += reveal_bit() / 2^k
  while (I[l+1] <= x) :
    l += 1
  if (x + 1/2^k < I[l+1] or I[l+1] == 1) :
      return k
  k += 1
```

Figure 2: Algorithm 1. Here $I[m] = m/N = p_1 + \ldots + p_m$.

$$0.x_1 \ldots x_k = \frac{1}{2^k}\left(\left\lceil \frac{2^k m}{N} \right\rceil - 1\right) .$$

The set of numbers in $[0, 1]$ starting with those exact first k digits is

$$\left[\frac{1}{2^k}\left(\left\lceil \frac{2^k m}{N} \right\rceil - 1\right), \frac{1}{2^k}\left\lceil \frac{2^k m}{N} \right\rceil\right),$$

therefore we conclude that

$$\{x \in [0, 1] : T(x) > k\} = \bigcup_{m=1}^{N-1} \left[\frac{1}{2^k}\left(\left\lceil \frac{2^k m}{N} \right\rceil - 1\right), \frac{1}{2^k}\left\lceil \frac{2^k m}{N} \right\rceil\right] .$$

Since $2^{-k} < 1/N$ each term of the union is disjoint and so

$$\mathbb{P}\left(T > k\right) = \frac{N-1}{2^k} ,$$

where $\{\cdot\}$ denotes the fractional part.

Therefore

$$\mathbb{E}[T] = 1 + \lfloor \log_2(N-1) \rfloor + \sum_{k: N \leq 2^k} \left(\frac{N-1}{2^k}\right) = 1 + \lfloor \log_2(N-1) \rfloor + \frac{N-1}{2^{\lfloor \log_2(N-1) \rfloor}} .$$

In all we deduce that the redundancy $\mathbb{E}[T] - H(Y)$ satisfies

$$R(x) = 2^x - x + 1 - \log_2\left(1 + \tfrac{1}{N-1}\right) ,$$

where $x = \{\log_2(N-1)\}$, where $\{\cdot\}$ denotes the fractional part.

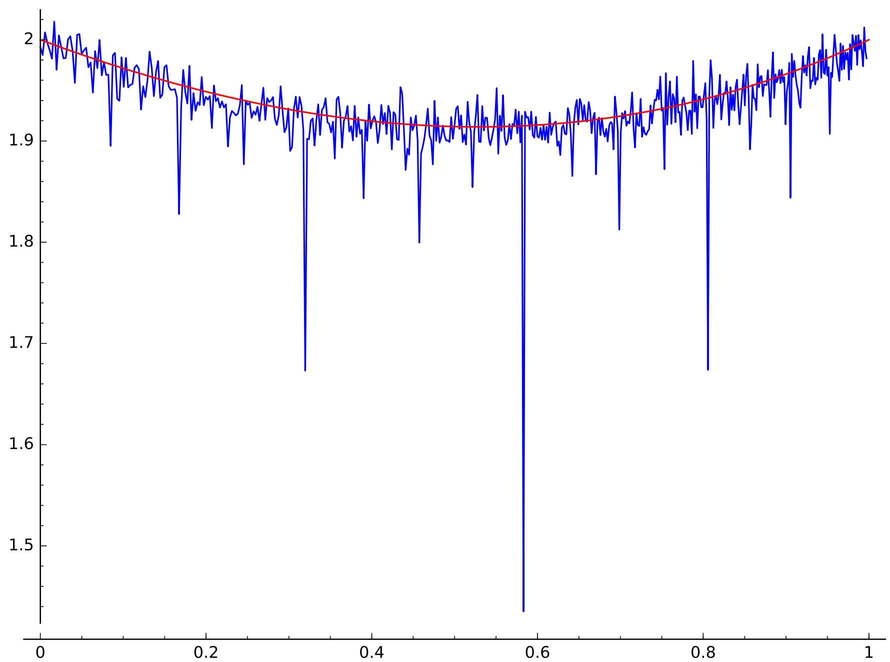

Figure 3: Experimental redundancy for $j = 9$.

Algorithm 2. In our second algorithm we stop as soon as $(0.x_1 \dots x_k, 0.x_1 \dots x_k + 2^{-k}) \subset I_j$ for some $j = 1, \dots, N$.

In this case we observe that the difference occurs when $0.x_1 \dots x_k + 2^{-k} = m/N$ for some $1 \leq m \leq n-1$. Of course $0.x_1 \dots x_k + 2^{-k} = A/2^B$ for some integers $A, B > 0$, with A odd.

This never happens when N is odd, therefore the expected value remains the same when N is odd. Now assume $N = 2^\nu d$ where d is odd. Then $m = 2^{\nu - B} dA$ and since A and d are odd we must have $B \leq \nu$ and $A < 2^B$. Now given $0 < A/2^B < 1$, what is the smallest k for which $0.x_1 \dots x_k + 2^{-k} = A/2^B$ occurs? Let us remark that, since $0.x_1 \dots x_k + 2^{-k} < 1$, all of the digits $x_1, \dots, x_k$ cannot equal 1, therefore $0.x_1 \dots x_k + 2^{-k}$ can be reduced to a number of the form $0.y_1 \dots y_k$ with k digits. It follows then that $k = B$ and that $m/N = A/2^B$ occurs as a right border for $k = B-1$. Observe that $2^{B-1} < 2^\nu \leq N$, hence, in fact, this happens during the phase in which $1/N < 2^{-k}$ and the intervals $(0.x_1 \dots x_k, 0.x_1 \dots x_k + 2^{-k})$ are longer than the intervals $[m/N, (m+1)/N]$, therefore once we get to the phase $1/N \geq 2^{-k}$, all of the possible dyadics are already available.

Fixed $1 \leq B \leq \nu$ there are $2^B - 2^{B-1}$ choices for our odd A (if we allow for the possibility of $m/N = 1$, whihc we have to discount anyway). Hence in all we can produce 2^ν dyadics as a right-border.

Therefore

$$\mathbb{E}[T] = 1 + \lfloor \log_2(N-1) \rfloor + \sum_{k: N \le 2^k} \left(\frac{N - 2^\nu}{2^k} \right) = 1 + \lfloor \log_2(N-1) \rfloor + \frac{N - 2^\nu}{2^{\lfloor \log_2(N-1) \rfloor}} .$$

In all we deduce that the redundancy $\mathbb{E}[T] - H(Y)$ satisfies

$$R(x) = 2^x - x + 1 - \frac{2^{\nu(N)} - 1}{N-1} 2^x - \log_2 \left(1 + \tfrac{1}{N-1} \right) ,$$

where $x = \{\log_2(N-1)\}$, where $\{\cdot\}$ denotes the fractional part and $\nu(N)$ is the greatest t such that 2^t divides N.

1.2 Generic distribution

Algorithm 1 for a generic Y. In this case we have to work with each individual right-border $P(j) := p_1 + \ldots + p_j$. As long as $p_j \le 2^{-k}$, all of the points in $x \in [P(j-1), P(j))$ will produce a truncation $0.x_1 \ldots x_k$ that satisfies $[0.x_1 \ldots x_k, 0.x_1 \ldots x_k + 2^{-k}] \not\subset [P(j-1), P(j))$, thus we will not have stopped. This accounts for a term $p_j \left(1 + \lfloor \log_2 (1/p_j) \rfloor\right)$ in $\sum_{k \ge 0} \mathbb{P}(T > k)$, for each $j \le N$ (even for $j = N$).

In particular this means that

$$H(Y) = \sum_{j=1}^{N} p_j \log_2 (1/p_j) \le \sum_{j=1}^{N} p_j \left(1 + \lfloor \log_2 (1/p_j) \rfloor\right) \le \mathbb{E}[T] .$$

The other cases are counted in (not necessarily disjoint from the previous count!)

$$U_k(\mathbf{p}) := \bigcup_{\substack{m=1, \\ 2^k > 1/p_m}}^{N-1} \left[\frac{1}{2^k} \left(\lceil 2^k P(m) \rceil - 1 \right), \frac{1}{2^k} \lceil 2^k P(m) \rceil \right] ,$$

which has measure

$$\mu_k(\mathbf{p}) := \frac{\#\{m \in \{1, \ldots, N-1\} : 2^k > 1/p_m\}}{2^k} .$$

Strictly speaking we should count the difference

$$|U_k(\mathbf{p}) \setminus V_k(\mathbf{p})| , \qquad V_k(\mathbf{p}) := \bigcup_{\substack{m=1, \\ 2^k \le 1/p_m}}^{N} \Big[P(m-1), P(m) \Big] .$$

We prove the bound

$$\sum_k \mu_k(\mathbf{p}) \leq 2\,.$$

Indeed, observe that

$$\sum_k \mu_k(\mathbf{p}) = \sum_k \sum_{m<N:p_m>\frac{1}{2^k}} \frac{1}{2^k}$$

and reverse the order of summation to get

$$\sum_k \mu_k(\mathbf{p}) = \sum_{m<N} \sum_{k:p_m>\frac{1}{2^k}} \frac{1}{2^k} = \sum_{m<N} \frac{1}{2^{\lfloor \log_2(1/p_m)\rfloor}} \leq 2\sum_m p_m = 2\,.$$

In general this means that

$$\mathbb{E}[T] \leq 1 + \sum_{j=1}^{N} p_j \lfloor \log_2(1/p_j)\rfloor + \sum_{1\leq m<N} \frac{1}{2^{\lfloor \log_2(1/p_m)\rfloor}}\,, \tag{1}$$

and in particular

$$\mathbb{E}[T] \leq H(Y) + 3\,.$$

Lemma 1 *Fix $p \in (0,1)$, then*

$$p\lfloor \log_2(1/p)\rfloor + \frac{1}{2^{\lfloor \log_2(1/p)\rfloor}} \leq p\log_2(1/p) + p$$

PROOF: Write

$$\log_2(1/p) = \lfloor \log_2(1/p)\rfloor + \epsilon\,,$$

where of course $\epsilon \in [0,1)$. Then

$$p\lfloor \log_2(1/p)\rfloor + \frac{1}{2^{\lfloor \log_2(1/p)\rfloor}} = p(\log_2(1/p) - \epsilon) + p2^{\epsilon}$$

and then the inequality $2^\epsilon \leq 1+\epsilon$, valid for $\epsilon \in [0,1]$, proves the result. □

Comment. To prove that $2^\epsilon \leq 1+\epsilon$ for $\epsilon \in [0,1]$, observe that the function $f(x) = 1 + x - 2^x$ is concave and $f(0) = f(1) = 0$.

As a corollary we get the following Theorem

Theorem 2 *Let T be the number of bits it takes to decide to which interval I_j the number X belongs. Then we have*

$$H(Y) \leq \mathbb{E}[T] \leq H(Y) + 2\,, \tag{2}$$

where $H(Y) = \sum_{j=1}^{k} p_j \,\log_2(1/p_j)$ is called the entropy of Y.

Furthermore, the $+2$ in (2) is tight by our example with the uniform distribution.

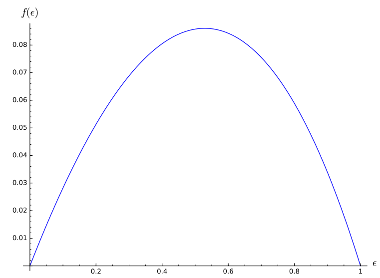

Figure 4: The function $f(\epsilon) = 1 + \epsilon - 2^{\epsilon}$.

Concluding remarks. I got the inspiration for this post from reading [1], chapter 5, where the optimal algorithm for producing a random variable from fair coin tosses is described. I wondered, usually when working with probabilities one uses the procedure that I explained in the post, which more or less corresponds to the so called *Inversion method*, so how far is it from the optimum? In this post I have proved that it is not that far from being optimal actually. In [1] it is proved that the optimal procedure satisfies the same bounds for the expected value $H(Y) \leq \cdot \leq H(Y) + 2$, but it is not shown that the inequality on the RHS is tight for the optimal procedure.

References

[1] Thomas M. Cover, Joy A. Thomas, *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing, Second Edition, 2006. 7