Quantum Bernoulli Factory

Suzie Brown

July 16, 2018

Bernoulli factories

Suppose we have access to a black box producing coin flips where the probability of observing heads is p. Roughly speaking, a Bernoulli factory is an algorithm that uses queries of this black box to produce a coin flip where the probability of observing heads is $f(p)$, for some specified function f.

Let us now make this notion more precise.

Definition 1. Let $f : S \to [0,1]$ be a function with domain $S \subseteq [0,1]$, and suppose we have a sequence of Bernoulli random variables $X_1, X_2, \ldots \overset{iid}{\sim} \text{Bernoulli}(p)$ with unknown parameter $p \in [0,1]$. Let $U \in \mathcal{U}$ denote a set of auxiliary random variables with known distributions, independent of p. Let τ be a stopping time with respect to the natural filtration, such that $\mathbb{P}(\tau < \infty) = 1$. A *Bernoulli factory* is a function $\mathcal{A} : \mathcal{U} \times \{0,1\}^T \to \{0,1\}$ such that $\mathcal{A}(U, X_1, \ldots, X_T) \sim \text{Bernoulli}(f(p))$ for all $p \in [0,1]$. For brevity we also denote $Y := \mathcal{A}(U, X_1, \ldots, X_T)$.

Example 1. $f(p) \equiv 1/2$.
The following solution to this problem is described in Von Neumann (1951). We flip the coin twice and, if the two outcomes are different, take the second one as the result. If the outcomes are the same, we start again. In this case no auxiliary random variable U is required.

$$\begin{aligned}\tau &= \min\{t \in \{2, 4, \ldots\} : X_{t-1} \neq X_t\} \\ \mathcal{A}(X_1, \ldots, X_T) &= X_\tau\end{aligned}$$

It is easy to show that this procedure produces heads with probability 1/2:

$$\mathbb{P}(Y = 1) = \mathbb{P}(X_\tau = 1) = \mathbb{P}(X_2 = 1 \mid X_2 \neq X_1) = \frac{\mathbb{P}(X_1 = 0, X_2 = 1)}{\mathbb{P}(X_1 = 0, X_2 = 1) + \mathbb{P}(X_1 = 1, X_2 = 0)} = \frac{p(1-p)}{2p(1-p)} = \frac{1}{2}$$

The running time τ of this Bernoulli factory is random and unbounded. In particular, $\tau \overset{d}{=} 2 \times \text{Geom}(2p(1-p))$. The expected running time is minimised when $p = 1/2$ (i.e. we already have a fair coin), where $\mathbb{E}(\tau) = 4$. This is why Von Neumann (1951) claims "the amended process is at most 25 percent as efficient as ordinary coin-tossing". He motivates this as a technique to ensure perfectly unbiased coin flips, although the more biased the original coin flipping procedure, the less efficient this correction will be, as illustrated in Figure 1.

Example 2. $f(p) = p^k, k \in \{1, 2, \ldots\}$.
This problem is solved easily by flipping the coin k times and outputting heads only if all k flips return heads. Again no auxiliary random variable is required.

$$\begin{aligned}\tau &= k \\ \mathcal{A}(X_1, \ldots, X_T) &= \mathbb{I}\{X_1 = 1, X_2 = 1, \ldots, X_\tau = 1\}\end{aligned}$$

In this case the running time is deterministically equal to k. It is clear that this procedure produces heads with probability p^k, since

$$\mathbb{P}(Y = 1) = \mathbb{P}(X_1 = 1, \ldots, X_k = 1) = \mathbb{P}(X_1 = 1) \ldots \mathbb{P}(X_k = 1) = \mathbb{P}(X_1 = 1)^k = p^k$$

The procedure can be sped up if we allow a random running time, still bounded above by k, by stopping the process early if tails comes up (in which case we know immediately that the output will be tails). In this case we have

$$\tau = k \wedge \min\{t \in \{1, 2, \ldots\} : X_t = 0\}$$

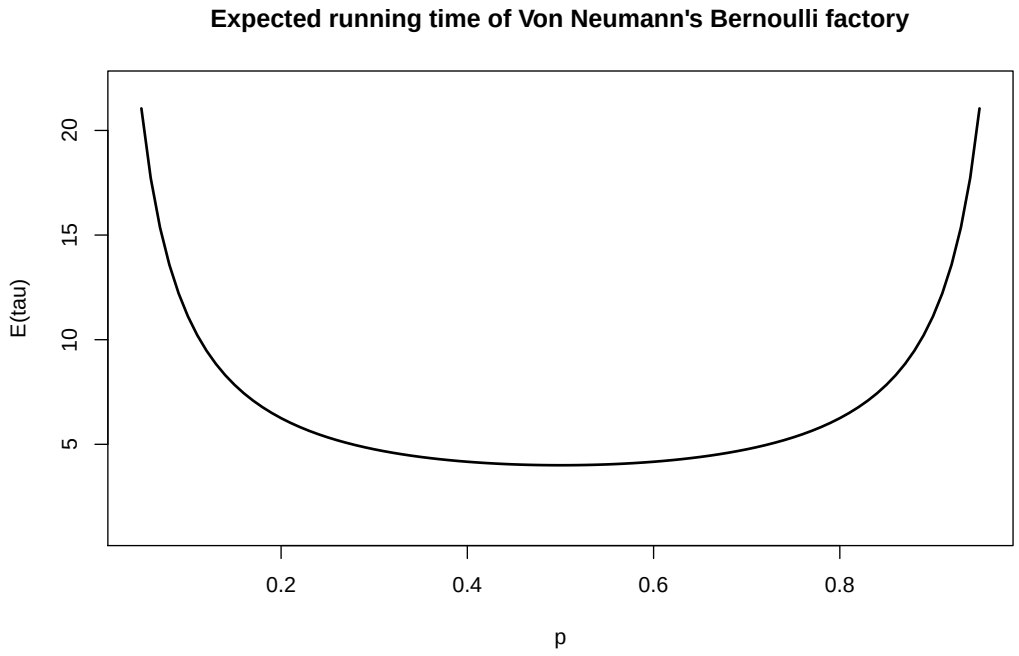

Figure 1: Expected running time of Von Neumann's $f(p) = 1/2$ procedure. When p is not too far from 1/2, the running time is moderate, but as p gets close to 0 or 1 the running time grows unboundedly. The expected running time achieves its minimum of four queries when $p = 1/2$.

and $\tau \stackrel{d}{=} \min\{k, \text{Geom}(1-p)\}$. Figure 2 shows the expected running time of this procedure for a few values of k. We see that early stopping can significantly reduce the expected running time when p is not close to 1, and that the gain in efficiency is greater for larger k. In the case $k = 1$, where f is the identity, the number of coin flips required is deterministically equal to one, for any p.

A natural extension of Example 2 is that products of simulable functions are themselves simulable. That is, given a Bernoulli factory for $f(p)$ and another for $g(p)$, it is possible to produce a Bernoulli factory for $f(p)g(p)$, by performing the two procedures one after the other and returning heads only if both return heads. As long as the procedures for f and g finish in finite time almost surely, then so will the fg procedure, and if the f and g procedures both have bounded running time then so will the fg procedure.

Similarly, we can obtain a Bernoulli factory for $(f \circ g)(p)$ by using outputs of the g procedure as inputs to the f procedure (provided the range of g is a subset of the domain of f). Even if both procedures have unbounded running times, the running time of the composition is almost surely finite.

Theorem 1. *Any constant function $f(p) \equiv c$ with $c \in [0,1] \cap \mathbb{Q}$, where c has a finite binary expansion, is strongly simulable in at most n queries to a (1/2)-coin, where 2^n is the denominator when c is expressed as a simplified fraction. Any real number $c \in [0,1]$ can be approximated arbitrarily well (by a strongly simulable function), and the approximation error decreases exponentially with the number of queries used.*

Proof. Consider the set of numbers c in (0,1) that can be expressed as finite binary decimals. Clearly, this excludes irrational numbers, but also excludes numbers like 1/3 which have a recurring binary expansion. Any element of this set can be expressed as a fraction $c = m/2^n$ where the n^{th} is the last non-zero entry in the binary expansion of c. Indeed this will be a simplified fraction, because if the numerator was divisible by two then the binary expansion would terminate one place earlier.

Therefore this set of numbers can be represented exactly by the set of fractions with an odd numerator and a power of 2 in the denominator. This representation does not double-count anything since it contains only simplified fractions (that is, fractions with an odd numerator), so no two can be equal.

We will now describe a constructive approach to simulate $f(p) \equiv c$ where $c = m/2^n$ is any of these numbers.

- If $2m > 2^n$, write your fraction as $1 - \frac{2^n - m}{2^n}$, and take $\frac{2^n - m}{2^n}$ as your new fraction, then proceed to the next step.

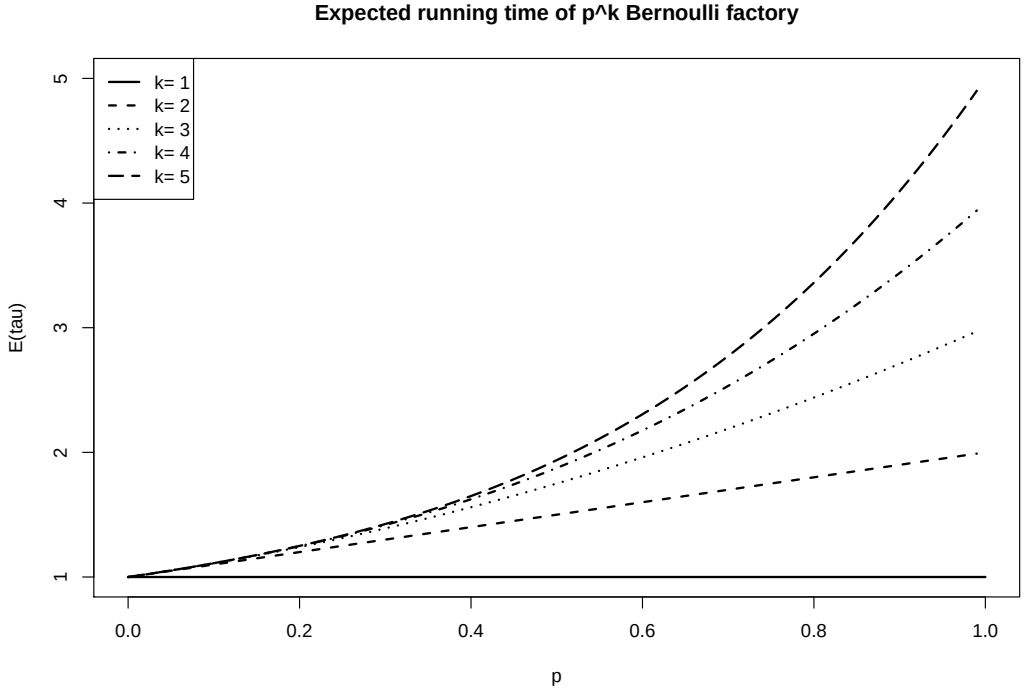

Figure 2: Expected running time for the $f(p) = p^k$ Bernoulli factory described, with early stopping. Without early stopping, the running time is deterministically equal to k. The early stopping procedure significantly increases the efficiency, particularly when p is small or k is large. The minimum number of queries is one for all k and p, and of course the $k = 1$ (identity) Bernoulli factory requires exactly one query for any p.

- Find the smallest k such that $2^k > m$, write your fraction as $\frac{1}{2^{n-k}}\frac{m}{2^k}$, and take $\frac{m}{2^k}$ as your new fraction.
- Repeat these steps until the fraction you are left with is $1/2$, then go back through the steps taken to write the fraction as a product of terms like $1/2^i$ and $(1 - 1/2^j)$. (The number of such terms is exactly the original n.)

Before showing that this procedure is correct, we illustrate it with an example. Suppose we have $c = 11/16$. Then the decomposition after each step is as follows:

$$\begin{aligned}\frac{11}{16} &= 1 - \frac{5}{16} = 1 - \frac{1}{2} \times \frac{5}{8} \\ &= 1 - \frac{1}{2}\left(1 - \frac{3}{8}\right) = 1 - \frac{1}{2}\left(1 - \frac{1}{2} \times \frac{3}{4}\right) \\ &= 1 - \frac{1}{2}\left(1 - \frac{3}{8}\right) = 1 - \frac{1}{2}\left(1 - \frac{1}{2} \times \frac{3}{4}\right) \\ &= 1 - \frac{1}{2}\left(1 - \frac{1}{2}\left(1 - \frac{1}{4}\right)\right) = 1 - \frac{1}{2}\left(1 - \frac{1}{2}\left(1 - \frac{1}{2} \times \frac{1}{2}\right)\right)\end{aligned}$$

Figure 3 depicts the resulting algorithm as a logic circuit, obtained by going through the final decomposition from right to left, where $\times$ is equivalent to an AND gate and $1-$ is equivalent to a NOT gate.

Since $1-$ operations leave the denominator invariant, the number of $(1/2)$-coin queries required is the number of $(1/2)$s that must be multiplied together to get the denominator 2^n, which is equal to n. The number of multiplications used and the number of inversions used are both $\leq n - 1$. The example in Figure 3 requires the maximal number of operations for a fraction with denominator 16.

An obvious extension of this result is that any constant $c \in [0, 1]$ can be simulated approximately. To simulate c to the nearest 2^{-n}, we round its binary expansion to n places and simulate the resulting fraction as above. Clearly we can simulate c with arbitrary small error. This procedure requires n queries of a $(1/2)$-coin and produces error at most 2^{-n-1}. Thus the error decays exponentially with the number of queries used. □

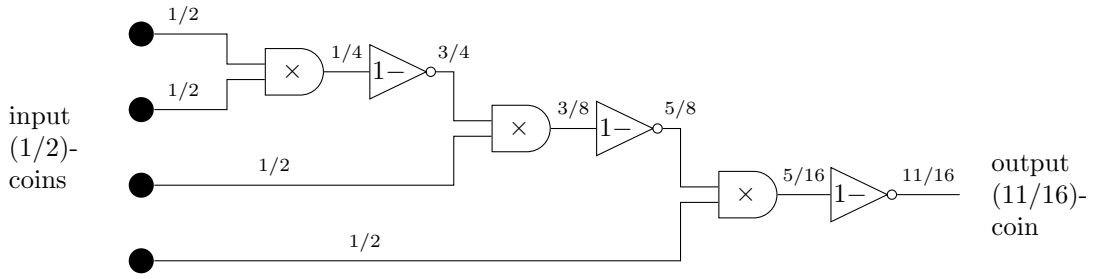

Figure 3: Logic circuit representation of an algorithm to produce one (11/16)-coin flip from four (1/2)-coin flips

It seems intuitively that, particularly for extreme values of p, this algorithm may be rather inefficient (reall that the expected number of queries of a p-coin to produce a (1/2)-coin goes to infinity as p approaches 0 or 1). Perhaps we can come up with an algorithm that is more robust, not relying on simulating (1/2)-coins?

References

Łatuszyński, K., Kosmidis, I., Papaspiliopoulos, O. and Roberts, G. O. (2011), 'Simulating events of unknown probabilities via reverse time martingales', *Random Structures & Algorithms* **38**(4), 441–452.

Von Neumann, J. (1951), '13. various techniques used in connection with random digits', *Appl. Math Ser* **12**(36-38), 3.