Sampling Uniformly from the Unit Simplex

Noah A. Smith and Roy W. Tromble
Department of Computer Science /
Center for Language and Speech Processing
Johns Hopkins University
{nasmith, royt}@cs.jhu.edu

August 2004

Abstract

We address the problem of selecting a point from a unit simplex, uniformly at random. This problem is important, for instance, when random multinomial probability distributions are required. We show that a previously proposed algorithm is incorrect, and demonstrate a corrected algorithm.

1 Introduction

Suppose we wish to select a multinomial distribution over n events, and we wish to do so uniformly across the space of such distributions. Such a distribution is characterized by a vector $\vec{p} \in \mathbb{R}^n$ such that

$$\sum_{i=1}^{n} p_i = 1 \tag{1}$$

and

$$p_i \geq 0, \forall i \in \{1, 2, ..., n\} \tag{2}$$

In practice, of course, we cannot sample from $\mathbb{R}^n$ or even an interval in $\mathbb{R}$; computers have only finite precision. One familiar technique for random generation in real intervals is to select a random integer and normalize it within the desired interval. This easily solves the problem when $n = 2$; select an integer x uniformly from among $\{0, 1, 2, ..., M\}$ (where M is, perhaps, the largest integer that can be represented), and then let $p_1 = \frac{x}{M}$ and $p_2 = 1 - \frac{x}{M}$.

What does it mean to sample uniformly under this kind of scheme? There are clearly $M + 1$ discrete distributions from which we sample, each corresponding to a choice of x. If we sample x uniformly from $\{0, 1, ..., M\}$, then then we have equal probability of choosing any of these $M + 1$ distributions.

Our goal is to generalize this technique for arbitrary n, maintaining the property that each possible distribution—i.e., those that are possible where we normalize by M so that every p_i is some multiple of $\frac{1}{M}$—gets equal probability.

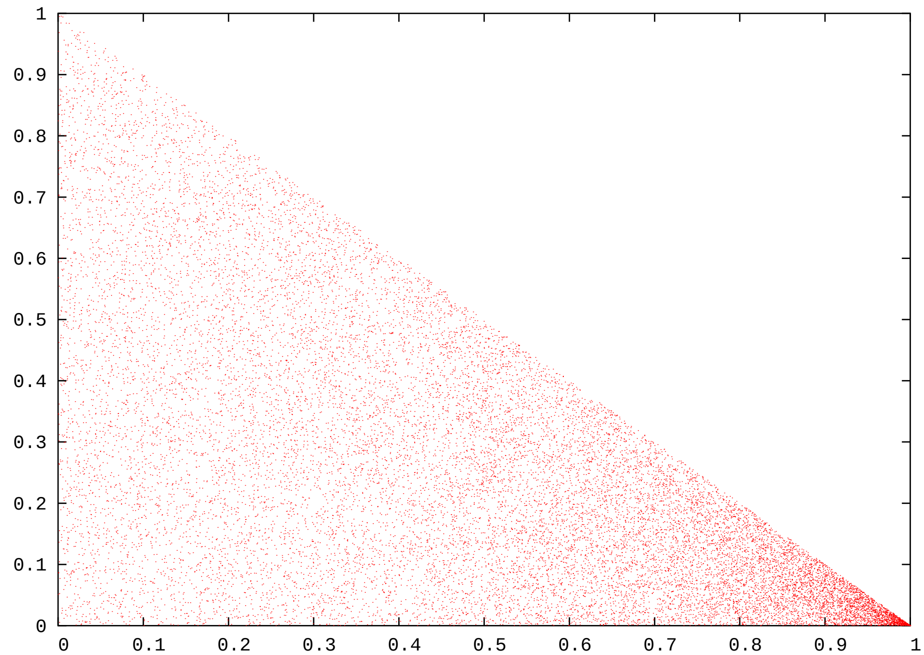

Figure 1: Obvious non-uniformity of sampling in the $n = 3$ case of the first naïve algorithm. The points are (p_1, p_2); p_3 need not be shown. 20,000 points were sampled.

2 Naïve Algorithms

One naïve algorithm is as follows.[1] Select a sequence $a_1, a_2, ...a_{n-1}$, each uniformly at random from $[0, 1]$. Let

$$p_i = a_i \prod j = 1^{i-1}(1 - a_j), \forall i = \{1, 2, ...n - 1\} \tag{3}$$

and let $p_n = 1 - \sum_{i=1}^{n-1} p_i$. This is certainly a generalization of the $n = 2$ algorithm, but a simple experiment shows that the sampling is not uniform (see Figure 1).

It is worth pointing out also that this algorithm differs from the set-up described in the introduction, where each p_i is a multiple of $\frac{1}{M}$. If each a_i were chosen by sampling from $\{0, 1, ..., M\}$, then we would have p_i be a multiple of $\frac{1}{M^i}$, which suggests *a priori* that there is non-uniformity in the sampling (each p_i comes from a different domain).

A second naïve algorithm (also due to Weisstein) is to sample $a_1, a_2, ..., a_n$ each from $[0, 1]$ uniformly (using the given procedure) and then normalize them. This is also incorrect (see Figure 2), though the points will all be multiples of $\frac{1}{M \sum_{i=1}^{n} x_i}$ (where $p_i = \frac{x_i}{M}$).

3 Hypercube Method

Weisstein goes on to suggest a kind of importance sampling, where points are picked uniformly from a wider region for which uniform sampling can be done straightforwardly.[2] If the point happens to

[1] Eric W. Weisstein. "Triangle Point Picking." From *MathWorld—A Wolfram Web Resource*. `http://mathworld.wolfram.com/TrianglePointPicking.html`.

[2] Weisstein's article deals with picking a point in an arbitrary triangle; he uses an enclosing quadrilateral.

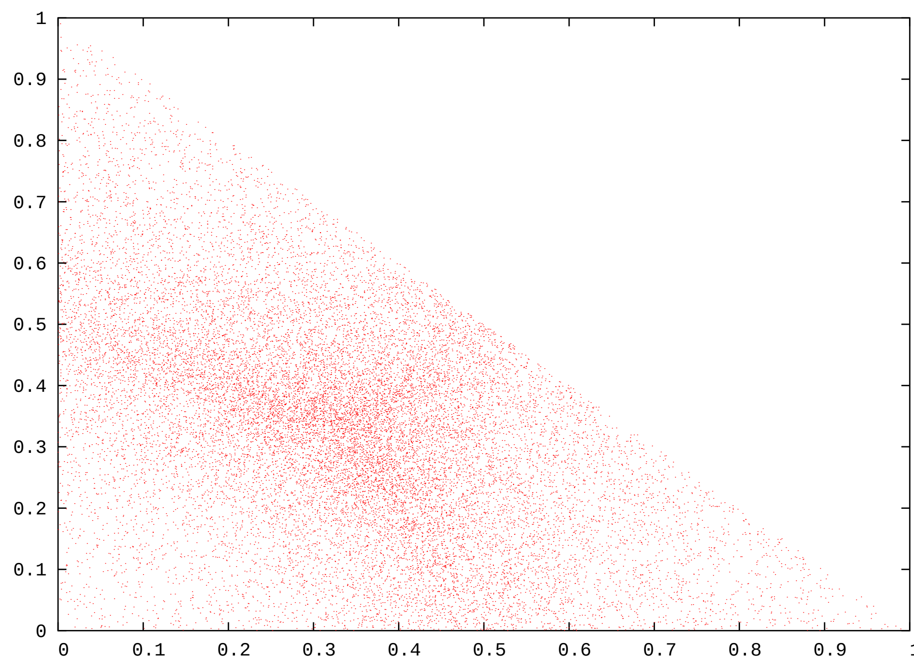

Figure 2: Obvious non-uniformity of sampling in the $n = 3$ case of the second naïve algorithm. The points are (p_1, p_2); p_3 need not be shown. 20,000 points were sampled.

fall outside the desired simplex, they can either be dropped or transformed via a function whose range is the simplex. This method is correct for the triangle point picking problem, where the transformation reflects a point across the side of the triangle that is internal to the quadrilateral.

Note that sampling uniformly from the unit simplex in $\mathbb{R}^n$ is equivalent to sampling uniformly from the set in $\mathbb{R}^{n-1}$ such that

$$\sum_{i=1}^{n-1} p_i \leq 1 \tag{4}$$

We can then choose $p_n = 1 - \sum_{i=1}^{n-1} p_i$.

Note that this new simplex (call it $\mathbb{S}^{n-1}$) is a subset of the unit hypercube in $\mathbb{R}^{n-1}$.

So we could sample each $p_i, \forall i \in \{1, 2, ..., n-1\}$ from $[0, 1]$ (i.e., sample uniformly from the unit hypercube). If $\sum_{i=1}^{n-1} > 1$, then we either reject the sample and try again, or transform it back into $\mathbb{S}^{n-1}$.

Rejection will become intractable as n increases, because the number of points in $\mathbb{S}^{n-1}$ as a fraction of the number of points in the $\mathbb{R}^{n-1}$ hypercube shrinks exponentially in n.

The question we seek to answer is, does there exist a transformation on points in the unit hypercube that maps evenly across $\mathbb{S}^{n-1}$? The next section describes a proposed mapping, and shows how it is not equivalent to uniform sampling. We then go on to give a mapping that is.

4 Kraemer Algorithm

The following algorithm is the only one we were able to find proposed for this problem.[3] First, select $x_1, x_2, ..., x_{n-1}$ each uniformly at random from $\{0, 1, ..., M\}$. Next, sort the x_i in place. Let $x_0 = 0$ and $x_n = M$. Now we have

$$0 = x_1 \leq x_2 \leq x_3 \leq ... \leq x_{n-1} \leq x_n = M \tag{5}$$

Let $y_i = x_i - x_{i-1}, \forall i \in \{1, 2, ..., n\}$. Now $\vec{y}$ will have the property that $\sum_{i=1}^{n} y_i = M$. Dividing by M will give a point in the unit simplex.

4.1 Incorrectness Proof

Under our assumption that we generate random reals from random integers, the above algorithm can be viewed as choosing $n - 1$ random integers and then deterministically mapping that vector to a vector of n integers that sum to M. By normalizing, we get a point in the unit simplex.

The mapping is a function f from a discrete set of $(M + 1)^{n-1}$ elements to a set of fewer elements (the set of points in the unit simplex where all coordinates are multiples of $\frac{1}{M}$). Call the range of f $\mathbb{T}^n$. For the sampling to be uniform across $\mathbb{T}^n$, we must verify that the $(M + 1)^{n-1}$ elements in the domain are equally distributed among all elements of $\mathbb{T}^n$. I.e.,

$$\left|\left\{\vec{x} : \vec{x} \in \{0, 1, ..., M\}^{n-1}, f(\vec{x}) = \vec{y}\right\}\right| = \frac{(M + 1)^{n-1}}{|\mathbb{T}^n|} \tag{6}$$

Suppose that we choose $\vec{x}$ and all elements are distinct and do not include 0 or M. How many $\vec{x'}$ will map to $f(\vec{x})$? The answer is that we must choose exactly the same set of coordinates of $\vec{x}$, but in any order. The number of $\vec{x'}$ that are permutations of $\vec{x}$, where all elements of $\vec{x}$ are distinct, is $(n - 1)!$. So the number of elements mapping to any $\vec{y} \in \mathbb{T}^n$ should be $(n - 1)!$.

Now consider $\vec{x}$ where two elements are identical. (This will result in a single p_i, apart from p_0 and p_n, begin zero.) How many $\vec{x'}$ will map to $f(\vec{x})$? The answer is of course the number of distinct permutations of $\vec{x}$, of which there are $\frac{(n-1)!}{2}$.

Generally speaking, the more zeroes present in a given $\vec{y}$, the lower the probability alloted to it under this sampling scheme. (There is a minor asymmetry about this; zeroes in the first and last positions of $\vec{y}$ do not cost anything.)

4.2 A Modification

If we are willing to eliminate all zeroes from the vector $\vec{y} \in \mathbb{T}^n$, a simple algorithm presents itself. Sample $x_1, ..., x_{n-1}$ uniformly at random from $\{1, 2, ..., M - 1\}$ *without replacement* (i.e., choose $n - 1$ distinct values). Let $x_0 = 0, x_n = M$. Let $y_i = x_i - x_{i-1}, \forall i \in \{1, 2, ..., n\}$.

Because each $\vec{x}$ contains all unique entries, we know that the equivalence classes mapping to the same $f(\vec{x})$ each contain exactly $(n-1)!$ vectors. So the sampling is uniform across distributions that have full support and where all p_i are multiples of $\frac{1}{M}$.

4.3 Allowing Zeroes

To equally distribute to the cases where some y_i are zero, as well, apply the above, no-zeroes algorithm with $n' = n, M' = M + n$. Then let $y_i = \langle f(\vec{x}) \rangle_i \vec{y} - 1$. Divide by M to normalize.

[3]This is due to Horst Kraemer's posting on the MathForum on December 20, 1999. `http://mathforum.org/epigone/sci.stat.math/quulswikherm/385e91a8.87536387@news.btx.dtag.de`.

4.4 Computational requirements

We assume that picking a random integer in $\{1, 2, ..., M+n\}$ is a constant-time operation. We also assume that a perfect hash function is available to ensure that no two coordinates of $\vec{x}$ are equal. Because we might choose a value already chosen, sampling might require more than 1 pick per x_i.

Suppose we are picking x_i. We have already selected $x_1, x_2, ..., x_{i-1}$. If we do importance sampling (i.e., pick x_i from $\{1, 2, ..., M+n-1\}$ and repeatedly reject until $x_i \notin \{x_1, x_2, ..., x_{i-1}\}$), then the expected runtime for generating x_i is given by r_i, where

$$r_i = \underbrace{\frac{M+n-i}{M+n-1}\cdot 1}_{\text{pick a novel value on the first try}} + \underbrace{\frac{i-1}{M+n-1}(r_i+1)}_{\text{fail and try again}} \tag{7}$$

$$= \frac{M+n-1}{M+n-i} \tag{8}$$

Summing over all i, we have a total expected time for sampling at:

$$\sum_{i=1}^{n-1} \frac{M+n-1}{M+n-i} \tag{9}$$

$$= (M+n-1)\sum_{i=1}^{n-1} \frac{1}{M+n-i} \tag{10}$$

$$= (M+n-1)(H_{M+1} - H_{M+n}) \tag{11}$$

Using bounds given by Young,[4] we can set the expected runtime for the sampling stage to be less than

$$(M+n-1)\left(\frac{1}{2(M+1)} - \frac{1}{2(M+n-1)} + \ln\left(\frac{M+1}{M+n}\right)\right) = O(n) \tag{13}$$

The sorting step can be done in $O(n \log n)$ steps. Overall runtime is therefore expected to be $O(n \log n)$.

The algorithm requires $O(n)$ space.

5 Conclusion

We have shown how triangle point picking algorithms do not generalize to uniform sampling from the unit simplex. We have discussed a previously proposed algorithm for this problem and demonstrated that it is incorrect. We have proposed an $O(n \log n)$ expected runtime, $O(n)$ space algorithm and demonstrated its correctness.

[4]Young, R. M. "Euler's Constant." *Math. Gaz.* 75, 187–190, 1991. See also `http://mathworld.wolfram.com/HarmonicNumber.html`.

$$\frac{1}{2(n+1)} + \ln n + \gamma < H_n < \frac{1}{2n} + \ln n + \gamma \tag{12}$$

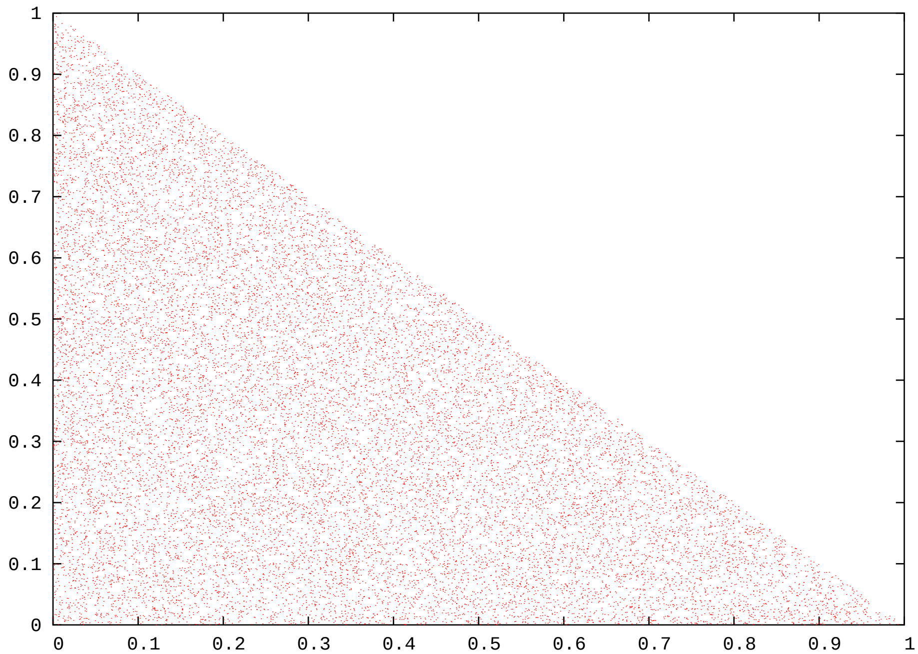

Figure 3: Our algorithm, $n = 3$. The points are (p_1, p_2); p_3 need not be shown. 20,000 points were sampled.