SAMPLING LIPSCHITZ CONTINUOUS DENSITIES

OLIVIER BINETTE

Abstract. A simple and efficient algorithm for generating random variates from the class of Lipschitz continuous densities is described. A MatLab implementation is freely available on GitHub (repository olivierbinette/lipsample).

1. Introduction

Let f be an integrable non-negative function on the interval $[0, 1]$. Given an upper bound M on f, the following acceptance-rejection algorithm generates a random variate following the density proportional to f.

Algorithm 0.

1. *Generate* (x, y) *uniformly distributed on the cube* $[0, 1] \times [0, M]$.
2. *If* $y > f(x)$*, then go to 1; otherwise return* x*.*

This is the simplest example of a *universal* or *black box* generator. It requires a function f, which may be evaluated at arbitrary points and a known upper bound M on f, but otherwise is not tailored to any particular functional form.

Here we refine Algorithm 0 in the case where f is Lipschitz continuous of a known order. The idea is to use the known Lipschitz order and the evaluation of f on a grid of points to construct a first-degree spline envelope which is then applied to acceptance-rejection sampling. Classes of densities with known Lipschitz continuity order include polynomial and trigonometric polynomial densities, log-polynomial densities such as the Von Mises and Gaussian densities, etc.

This work was motivated by poor simulation algorithms proposed in the litterature for particular classes of densities. For instance, Fernandez-Duran (2014) proposed the use of Algorithm 0 to simulate from the class of trigonometric polynomial densities. Since the first derivate of such densities is easily bounded, our method is an immediate (and rather significative) improvement.

1.1. Short litterature review.

Hörmann et al. (2004) describe universal generators for different classes of densities excluding Lipschitz continuous densities. Beliakov (2005) implements a universal generator for multivariate Lipschitz continuous densities based on zeroth degree spline approximate envelopes.

Date: December 8, 2017.

1.2. **Structure of the paper.** The rather simple method, which we refer to as the *LipSample* algorithm, is presented in section 2. Section 3 presents an efficient vectorized implementation in MatLab.

2. The LipSample algorithm

Recall that a function f defined on the interval $[0,1]$ is said to be of Lipschitz order M, denoted $f \in \mathrm{Lip}(M)$, if $\sup_{x,y\in[0,1]} \frac{f(x)-f(y)}{x-y} \leq M$. If f is continuously differentiable, then $f \in \mathrm{Lip}(\sup|f'|)$. The following algorithm generates random variables following the density proportional to $f \in \mathrm{Lip}(M)$.

Algorithm 1 (LipSample).

1. *Construct a spline envelope S of f.*
2. *Generate x following the density proportional to S and generate y uniformly on $[0, S(x)]$.*
3. *If $y > f(x)$, then go to 2; otherwise return x.*

We may also construct a lower spline envelope of f in order to accelerate step 3. Next we describe how to perform steps 1 and 2.

2.1. **The spline envelope.** For simplicity and computational efficiency, we use the first order spline envelope $S = S_n$ given by

$$S_n = \sum_{i=1}^{n} \left(f(\tfrac{i}{n}) + h_{i,n}\right) P_{i,n}, \tag{1}$$

where $h_{i,n} \in \mathbb{R}$ is to be determined, $P_{i,n}(x) = c_{i,n} \max\{0, 1 - |i - nx|\}$, $c_{i,n} = 2$ if $i = 0$ or $i = n$, $c_{i,n} = 1$ otherwise. Hence S_n linearly interpolates between the points $(i/n, f(i/n) + h_{i,n})$, $i = 0, 1, ..., n$.

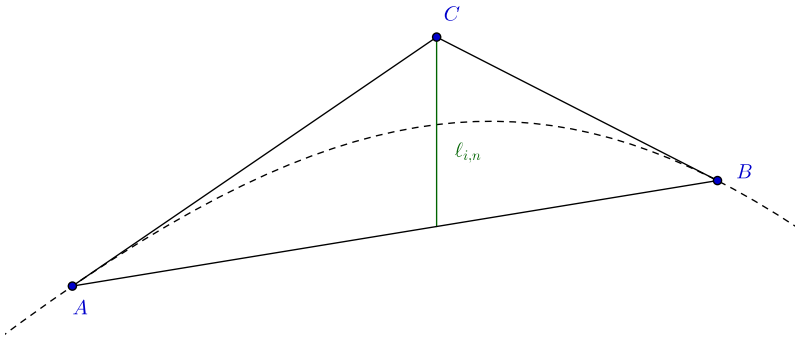

Figure 1

The constants $h_{i,n}$ must now be chosen as to ensure $S \geq f$. The simplest choice would be $h_{i,n} = L/(2n)$, but we can do better. Given two sample points $A = (\frac{i-1}{n}, f(\frac{i-1}{n}))$ and $B = (\frac{i}{n}, f(\frac{i}{n}))$, the Lipschitz condition on f yields a triangle ABC such that if $x \in [\frac{i-1}{n}, \frac{i}{n}]$ and $f(x)$ is greater than the value of the linear

interpolant of A and B at x, then $(x, f(x))$ is inside of ABC. Precisely, the slope of the line AC is $\arctan(M)$ and the slope of BC is $-\arctan(M)$. If say $C = (z, f(z))$ then we denote the difference between $f(z)$ and the value of the linear interpolant at z by $\ell_{i,n}$ (see Figure 1). It can be shown by elementary geometry that

$$\ell_{i,n} = r(M - n|(f(\tfrac{i}{n}) - f(\tfrac{i-1}{n})|), \tag{2}$$

where $r = \frac{1}{2}\|A-B\|\frac{\sin(\pi-\alpha-\beta)}{\sin(\alpha)}$, $\alpha = \arctan(M)$ and $\beta = \arctan(n|(f(\frac{i}{n})-f(\frac{i-1}{n})|)$. Hence taking

$$h_{i,n} = \max\{\ell_{i,n}, \ell_{i+1,n}\} \tag{3}$$

ensures that $S_n \geq f$ and furthermore this bound is sharp over $\mathrm{Lip}(M)$.

Note also that at any point $x \in [0,1]$, only two terms of the sum $S_n(x)$ are non-zero. In fact, letting $i = \lfloor (n+1)x \rfloor$, we find

$$S_n(x) = (nx - i)f(\tfrac{i}{n}) + (i + 1 - nx)f(\tfrac{i+1}{n}).$$

It is also quite easy to sample from the density proportional to S_n as it is a mixture of triangular densities.

Lemma 1. *Let $p_i = f(\frac{i}{n})+h_{i,n}$ if $i = 1, 2, ..., n-1$, $p_i = (f(\frac{i}{n})+h_{i,n})/2$ if $i = 0, n$. Suppose I is distributed as $P(I = i) \propto p_i$ and that $U_1, U_2 \sim^{ind.} unif([0,1])$. Let $Z = (U_1 + U_2 + I - 1)/n$ and*

$$X = |Z| + 2(1 - Z)\,\mathbb{I}_{Z>1}.$$

Then X is distributed proportionally to S_n.

Proof. Write $S_n = \sum_{i=0}^{n} p_i \tilde{P}_{i,n}$, where $\tilde{P}_{i,n} = P_{i,n}$ if $i = 1, 2, ..., n-1$ and $\tilde{P}_{i,n} = 2P_{i,n}$ if $i = 1, 2$. Finally, remark that $\int_{[0,1]} \tilde{P}_{i,n}$ does not depend on i and that conditionally on $I = i$, X is distributed proportionally to $\tilde{P}_i$. □

3. Implementation

Let the function f defined on $[0,1]$ be given together with its Lipschitz constant M. The following MatLab code produces random variates following the density proportional to f.

Step 1. Constructing the spline envelope.

```
% Number of components for the spline.
n = ceil(40*sqrt(M));

% Evaluate f at n+1 equidistant points and add the constants h_{i,n} = M / (2n)
x = linspace(0,1, n+1);
y = arrayfun(f, x) + M/(2*n);
```

Step 2. Generate random variates from the envelope.

```
% Assume nProp is given.
U1 = rand(1, nProp);
U2 = rand(1, nProp);

% Simulate from the discrete distribution specified by the weights y.
y(1) = y(1)/2;
y(end) = y(end)/2;
I = discretesample(y, nProp); % Function available on
    github.com/olivierbinette/LibSample.
y(1) = 2*y(1);
y(end) = 2*y(end);

% The random variates are given by U.
U = abs((U1 + U2 + I - 2)/n);
U(U > 1) = 2 - U(U > 1);
```

Step 3. Acceptance-rejection.

```
V = rand(1, nProp);
sample = U(lt(V .* interp1(x,y,U), B));
```

The preceding steps may be repeated if the size of the sample is smaller than needed.

3.1. **Improvements.** The following calculates the constants $h_{i,n}$ as in section 2 and may be used to replace Step 1.

```
x = linspace(0,1,n+1);
y = arrayfun(f, x);

% Use the Lipschitz constant to locally adjust the spline.
alpha = atan(L);
d = diff(y);
beta = abs(atan(n*d));
r = 0.5*sqrt(n^2 + d.^2).*sin(pi-alpha-beta)./sin(alpha);
ell = r.*(L - abs(n*d));

y(1) = y(1) + ell(1);
y(n+1) = y(n+1) + ell(n);
for i = 2:n
    y(i) = y(i) + max(ell(i-1), ell(i));
end
```

In our MatLab implementation available on Github (repository olivierbinette/LipSample), we also use a lower spline envelope to accelerate Step 3.

3.2. **Performance.** For the generation of a large number of normal variates restricted to $[-5, 5]$, the *LipSample* procedure is only about 50 times slower than MatLab's own *randn*. We consider this to be quite fast: *randn* has been internally

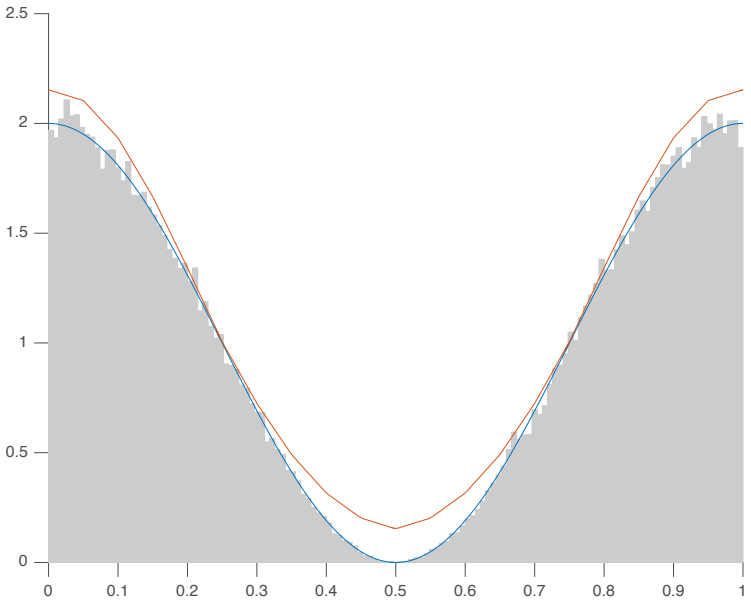

FIGURE 2. The density $f(x) = 1 + \cos(2\pi x)$ (blue), a spline envelope S_n (orange), and 100 000 random variates obtained using the LipSample algorithm (grey histogram).

optimized whereas our *LipSample* code is interpreted by MatLab and completely general.

REFERENCES

[1] Beliakov, G. (2005). *Class library ranlip for multivariate nonuniform random variate generation.* Computer Physics Communications, Volume 170, Issue 1.

[2] Devroye, L. (1988). Non-Uniform Random Variate Generation. *Springer-Verlag.*

[3] Fernández-Durán, J. and Gregorio-Domínguez, M. (2014). *Modeling angles in proteins and circular genomes using multivariate angular distributions based on multiple nonnegative trigonometric sums.* Statistical Applications in Genetics and Molecular Biology, 13(1), pp. 1-18.

[4] Hörmann, W., J. Leydold, J. and Derflinger, G. (2004). Automatic Nonuniform Random Variate Generation. *Springer.*

[5] Robert, C. et Casella, G. (2004). Monte Carlo Statistical Methods. *Springer-Verlag.*