Discrete Adaptive Rejection Sampling

Daniel R. Sheldon

September 30, 2013

Abstract

Adaptive rejection sampling (ARS) is an algorithm by Gilks and Wild for drawing samples from a continuous log-concave probability distribution with only black-box access to a function that computes the (unnormalized) density function. The ideas extend in a straightforward way to discrete log-concave distributions, but some details of the extension and its implementation can be tricky. This report provides the details of a discrete ARS algorithm. A companion implementation in C, with a MATLAB interface, accompanies the report.

1 Introduction

Adaptive rejection sampling (ARS) [Gilks and Wild, 1992] is a widely used technique for generating a real-valued random variable from a distribution with log-concave density function f. It requires only black-box access to f, and is based on the idea of upper bounding $\log f$, which is concave, with a piecewise linear function.

ARS belongs to a class of rejection sampling algorithms, many of which preceded ARS, that use piecwise linear upper bounds for concave log-density functions. Devroye [1986, 1987] describes several related techniques for sampling from log-concave distributions (continuous or discrete) that are *universally fast*, which means that their expected running time is constant. In contrast, ARS does not any running time guarantees.

Nevertheless, ARS is very popular. A likely reason is that ARS requires less prior knowledge about the probability distribution compared with the universally fast alternatives. ARS requires *only* black-box access to a function proportional to f. The universally fast methods typically require two additional pieces of information: the mode of f, and the normalization constant $Z = \int f$ (in other words, they require the density to be normalized).[1] In applications such as Gibbs sampling, for which ARS was originally developed, these quantities are rarely known. Thus, one can "begin sampling" immediately with ARS, while the universally fast methods would require additional setup steps to compute the mode and/or normalization constant. This property of ARS is very useful, and it is desirable to have an analogous sampler for discrete log-concave distributions.

In this report we provide the details of an adaptive rejection sampler for discrete log-concave distributions. A companion implementation in C, together with a MATLAB interface, accompanies the report. The report is self-contained. Section 2 provides the necessary background on discrete log-concave distributions and rejection sampling. Section 3 describes the adaptive rejection sampler.

2 Background

Let $p(k) = Z \cdot \Pr(X = k)$ be the unnormalized probability mass function of a discrete random variable X that is supported on the set of integers $\{L, L+1, \ldots, U\}$. Here, Z is an unknown normalization constant and we allow $L = -\infty$, $U = \infty$, or both.

[1] Devroye recently developed a universally fast method that works with unnormalized densities [Devroye, 2012]. The setup step involves a search which takes running time $O(|\log(Z/Z')|)$ where Z is the true normalization constant and Z' is a prior estimate.

We would like to generate a random variable from the distribution with mass function $p(\cdot)/Z$, given only black-box access to the function $p(\cdot)$ and the knowledge that $p(\cdot)$ is log-concave.

2.1 Log-concave Distributions.

A discrete distribution is log-concave if, for all k,

$$p(k)^2 \geq p(k-1)p(k+1). \tag{1}$$

This property is analagous to log-concavity of a density function f. Let $g(k) = \log p(k)$ be the log probability mass function, which is the function that we want to upper bound in discrete ARS. By rearranging (1), we see that

$$g(k) - g(k-1) \geq g(k+1) - g(k). \tag{2}$$

Define

$$m(k) = g(k+1) - g(k),$$

which can be interpreted as the slope of $g(\cdot)$ at k. Equation (2) shows that the sequence of slopes $\{m(k)\}$ is non-increasing, which is analogous to the condition that the second derivative of a log-density function f is negative.

The idea of adaptive rejection sampling in the continuous case is to upper-bound $\log f$ by tangent lines, which, due to concavity, always lie above $\log f$. The same idea works in the discrete case.

Proposition 1. *Fix any integer j (the "tangent point"), and let $m(j)$ be the slope as defined above. Then*

$$g(k) \leq g(j) + m(j)(k-j). \tag{3}$$

Proof. Let $k \geq j$. Then

$$\begin{aligned} g(k) &= g(j) + \sum_{i=j}^{k-1}(g(i+1) - g(i)) \\ &= g(j) + \sum_{i=j}^{k-1} m(i) \\ &\leq g(j) + m(j)(k-j). \end{aligned}$$

The case $k < j$ is similar. □

Thus, the linear function $h_j(k) = g(j) + m(j)(k-j)$, which is "tangent" to g at j, can be used to upper bound g. Discrete ARS uses the idea of upper bounding g with the pointwise minimum of a set of functions tangent to g at points $j_1, \ldots, j_n$.

2.2 Rejection Sampling.

Rejection sampling is a well-known technique for sampling from the distribution with mass function proportional to $p(\cdot)$ when one knows how to sample from the distribution with mass function proportional to $q(\cdot)$, where $q(k) \geq p(k)$ for all k. The simple procedure is shown in Algorithm 1.

To see that the random variable X generated by rejection sampling has a mass function proportional to p, note that the algorithm terminates when $U \cdot q(X) \leq p(X)$. Let $Z = \sum_k p(k)$ and $Z' = \sum_k q(k)$. The

Algorithm 1: Rejection sampling

```
Input  : Unnormalized mass function $p$, upper bound $q$
Result: A random variable $X$ from distribution $p/\sum p$
repeat
    Let $X$ be a random variable with distribution $q/\sum q$;
    Let $U$ be a uniform random variable;
until $U \le p(X)/q(X)$;
```

probability of termination a given iteration is:

$$\begin{aligned}\Pr(U \le p(X)/q(X)) &= \sum_k \Pr(X = k) \Pr(U \le p(k)/q(k)) \\ &= \sum_k \frac{q(k)}{Z'} \frac{p(k)}{q(k)} \\ &= \frac{1}{Z'} \sum_k p(k) = Z/Z'.\end{aligned}$$

This shows that expected number of iterations is Z'/Z. The conditional probability that $X = k$ given that the algorithm terminated in the current iteration is:

$$\begin{aligned}\Pr(X = k \mid U \le p(X)/q(X)) &= \frac{\Pr(X = k) \Pr(U \le p(X)/q(X) \mid X = k)}{\Pr(U \le p(X)/q(X))} \\ &= \frac{q(k)/Z' \cdot p(k)/q(k)}{Z/Z'} \\ &= \frac{p(k)}{Z}.\end{aligned}$$

This shows that X has the desired distribution.

3 Adaptive Rejection Sampling

There are two main ideas underlying adaptive rejection sampling. The first idea is to use a piecewise-linear upper bound h of the log probability mass function, which is easy to build using tangent lines, to construct an upper bound for rejection sampling. Specifically, we use the simple observation:

$$h(k) \ge \log p(k) \Rightarrow \exp h(k) \ge p(k).$$

If h is piecewise-linear, then $q = \exp h$ consists of pieces that are either exponential or uniform (when the tangent-line has slope zero), and samples from q can be generated efficiently.

The second idea is to improve the piecewise-linear upper-bound h adaptively during the sampling process. At all times, h will consist of pieces that are tangent to $\log p$ at points $j_1 < j_2 \ldots < j_n$. A small set of initial points is provided by the user, and then each time a point $X = j$ is rejected, it is added to list to improve the quality of the upper bound. Gilks and Wild [1992] observed empirically that this process (for continuous distributions) quickly improves the quality of the upper bounding function and leads to an efficient sampler.

The discrete ARS procedure is presented in Algorithm 2. The algorithm includes the so-called "squeeze test": if r is a lower-bound of p, then the condition $U \le r(X)/q(X)$ guarantees that $U \le p(X)/q(X)$, and, if $r(X)$ is much easier to compute than $p(X)$, leads to significant running-time savings whenever the first condition holds.

At a high level, discrete ARS is a straightforward extension to rejection sampling—the only new twists are that the upper bounding function changes in each iteration, and the squeeze test is performed. The remaining difficulties are:

```
Algorithm 2: Discrete Adaptive Rejection Sampling
  Input  : Black box access to an unnormalized mass function $p$, initial set of points $J$, number of
           samples $m$
  Output: An array of $m$ random numbers drawn from distribution $p/\sum p$

  Construct piecewise-exponential upper bound $q$ and lower bound $r$ using points $J$;
  while fewer than $m$ accepts do
      Generate $X$ with distribution $q/\sum q$;
      Generate uniform random variable $U$;
      // Squeeze test
      if $U \leq r(X)/q(X)$ then
          Accept $X$
      end
      // Normal rejection test
      else if $U \leq p(X)/q(X)$ then
          Accept $X$
      end
      else
          Add $X$ to $J$ and recompute upper and lower bounds $q$ and $r$
      end
  end
```

- How to construct the upper and lower bounds given the current set of points $j_1 < j_2 < \dots j_n$.
- How to sample from the piecewise-exponential upper bound.

We discuss each of these in turn.

3.1 The Upper and Lower Bounds

The upper and lower bounds to $g = \log p$ are both piecewise-linear. We focus here on the upper bound h, which consists of linear pieces $h_1, \dots, h_n$ parameterized as follows:

$$h_i(k) = m_i k + b_i, \quad k \in \{L_i, L_i + 1, \dots, U_i\}.$$

The pieces are ordered from left to right and cover the entire domain, i.e.,

$$L_1 = L, \quad U_i = L_{i+1} - 1, \quad U_n = U.$$

It is worth noting that most rejection methods for log-concave distributions use upper-bounds of this form, though not necessarily constructed using tangent lines as we will describe next. The sampling method that will appear in Section **??** is generic to any upper bounds of this form.

3.1.1 Constructing the Upper Bound Using Tangent Lines

We compute the parameters (m_i, b_i, L_i, U_i) of each piece by constructing tangent lines to $g = \log p$ at points $j_1 < \dots < j_n$. Recall that the tangent line at j_i is given by the equation:

$$h_i(k) = g(j_i) + \big(g(j_i + 1) - g(j_i)\big)(k - j_i).$$

Thus

$$m_i = g(j_i + 1) - g(j_i), \qquad b_i = g(j_i) - m_i j_i.$$

Algorithm 3: Construction of Upper Bounds

```
Input  : Points $J = \{j_1, \ldots, j_n\}$, function $g = \log p$
Output: Parameters $(m_i, b_i, L_i, U_i)$ of each piece
for $i = 1$ to $n$ do
    // Slope and intercept of $i$th piece
    $m_i = g(j_i + 1) - g(j_i)$;
    $b_i \leftarrow g(j_i) - m_i j_i$;
end
for $i = 1$ to $n - 1$ do
    // Intersection point between pieces $i$ and $i + 1$
    $x_i \leftarrow \frac{b_{i+1} - b_i}{m_i - m_{i+1}}$;
end
// Lower and upper limits for each piece
$(L_i, U_i) = \begin{cases} (L, \lfloor x_i \rfloor) & i = 1, \\ (\lceil x_{i-1} \rceil, \lfloor x_i \rfloor) & i = 2, \ldots, n-1, \\ (\lceil x_{i-1} \rceil, U) & i = n. \end{cases}$
```

Each tangent line h_i is a valid upper bound over the entire domain, but we restrict the piece to cover only the domain $\{L_i, \ldots, U_i\}$ for which h_i is the tightest (i.e. smallest) upper bound. In other words, the overall function h is the pointwise minimum of the tangent lines:

$$h(k) = \min_i h_i(k).$$

To compute the intervals over which each piece is minimum, we solve for the (real-valued) intersection point x_i between the tangent lines h_i and h_{i+1}, for $i = 1, \ldots, n - 1$. These satisfy

$$m_i x_i + b_i = m_{i+1} x_i + b_{i+1},$$

and thus

$$x_i = \frac{b_{i+1} - b_i}{m_i - m_{i+1}}.$$

The ith piece is minimum for all integers between x_{i-1} and x_i. Thus we set the bounds as follows:

$$(L_i, U_i) = \begin{cases} (L, \lfloor x_i \rfloor) & i = 1, \\ (\lceil x_{i-1} \rceil, \lfloor x_i \rfloor) & i = 2, \ldots, n-1, \\ (\lceil x_{i-1} \rceil, U) & i = n. \end{cases}$$

The entire procedure is summarized in Algorithm 3.

3.1.2 Constructing the Lower Bound Using Secant Lines

As a byproduct of rejection sampling, it is also easy to construct a valid piecewise linear lower bound of $g = \log p$. Consider the "secant" line r_i that agrees with g at points j_i and j_{i+1}:

$$\begin{aligned} r_i(k) &= \hat{m}_i k + \hat{b}_i, \\ \hat{m}_i &= \frac{g(j_{i+1}) - g(j_i)}{j_{i+1} - j_i}, \\ \hat{b}_i &= g(j_i) - m_i j_i. \end{aligned}$$

The reader can verify using log-concavity that $r_i(k) \leq g(k)$ for all $k \in \{j_i, \dots, j_{i+1}\}$. The overall lower bound is

$$r(k) = \begin{cases} -\infty & k < j_1 \text{ or } k > j_n, \\ r_i(k) & k \in \{j_i, \dots, j_{i+1}\} \end{cases}.$$

3.2 Sampling

The remaining part of the algorithm is to sample from the distribution $q = \exp h$.

3.2.1 The Inversion Method

We will use the well-known *inversion method.* The cumulative distribution function is:

$$F(k) = \sum_{i=L}^{k} q(i)/Z,$$

where $Z = \sum_{i=L}^{U} q(i)$ is the normalization constant. The inversion method works by generating a random variable U that is uniform on $[0,1]$, and then letting X be the smallest integer such that $F(X) > U$. To see that X has the correct distribution, observe that

$$\begin{aligned} \Pr(X = k) &= \Pr\big(F(k-1) \leq U < F(k)\big) \\ &= F(k) - F(k-1) \\ &= q(k)/Z. \end{aligned}$$

Remark. Define $F^{-1} : [0,1] \to \{L, \dots, U\}$ by letting $F^{-1}(p)$ be the smallest k such that $F(k) > p$. Then we obtain $X = F^{-1}(U)$ by *inverting* the cdf.

3.2.2 Inverting a Piecewise-Geometric Sequence

Because the pieces of q are either geometric or constant, the inversion can be done relatively easily. A primary operation we need is the ability to compute partial sums of a geometric sequence, so we start by stating a useful identity.

Proposition 2. *Let $q(k) = ce^{mk}$ for $k \in \{a, \dots, b\}$, with $m \neq 0$. Then*

$$\sum_{k=a}^{b} q(k) = \frac{q(a) - q(b+1)}{1 - e^m}. \tag{4}$$

Proof. This is a standard result about geometric sequences:

$$\sum_{k=a}^{b} q(k) = ce^{ma} \sum_{k=0}^{b-a} e^{mk} = \frac{ce^{ma} - ce^{m(b+1)}}{1 - e^m} = \frac{q(a) - q(b+1)}{1 - e^m}.$$

□

When $m > 0$, the numerator and denominator of (4) can both have large magnitudes, so it is best to multiply both by e^{-m} to avoid the numerical instability, which gives an equivalent identity.

Proposition 3. *Let $q(k) = ce^{mk}$ for $k \in \{a, \dots, b\}$, with $m \neq 0$. Then*

$$\sum_{k=a}^{b} q(k) = \frac{q(b) - q(a-1)}{1 - e^{-m}}. \tag{5}$$

Before we proceed with the inversion, we first need to compute the total mass $Z_i := \sum_{k=L_i}^{U_i} q(k)$ of each piece. We use Equations (4) and (5) together with the definition $q(k) = \exp(m_i k + b_i)$ for $k \in \{L_i, \ldots, U_i\}$, and a simple calcluation for the case $m_i = 0$:

$$Z_i = \sum_{k=L_i}^{U_i} q(k) = \begin{cases} \dfrac{\exp(m_i L_i + b_i) - \exp(m_i(U_i + 1) + b_i)}{1 - \exp(m_i)} & m_i < 0, \\ \exp(m_i L_i + b_i)(U_i - L_i + 1) & m_i = 0, \\ \dfrac{\exp(m_i U_i + b_i) - \exp(m_i(L_i - 1) + b_i)}{1 - \exp(-m_i)} & m_i > 0. \end{cases} \tag{6}$$

Now let $Z = Z_1 + \ldots + Z_n$ be the overall normalization constant.

To perform the inversion, we generate the uniform random variable U and solve for the smallest integer X such that

$$\sum_{k=L}^{X} q(k) > UZ.$$

Let i be the smallest integer such that $Z_1 + \ldots + Z_i > UZ$. Then X belongs to the ith piece, and we now must search within the ith piece to find the smallest integer X such that

$$\sum_{k=L_i}^{X} q(k) > U', \tag{7}$$

where $U' = UZ - (Z_1 + \ldots Z_{i-1})$ is the remainder.

First, consider the case when $m_i < 0$. By Proposition 2, Condition (7) is equivalent to

$$\frac{q(L_i) - q(X+1)}{1 - e^{m_i}} > U'. \tag{8}$$

Since $q(y) = \exp(m_i y + b_i)$ is also well-defined on the real numbers and the left side of (8) is monotone, we can find the smallest integer for which (8) holds by solving it as an *equality* for a real value $X' \in [L_i, U_i + 1]$, and then taking $X = \lceil X' \rceil$. The result is:

$$X = \left\lceil \frac{\log\big(q(L_i) - U'(1 - e^{m_i})\big) - b_i}{m_i} - 1 \right\rceil$$

Next, consider the case when $m_i > 0$. The computation is similar except that we use Proposition 3 instead to get the following version of (7):

$$\frac{q(L_i - 1) - q(X)}{1 - e^{-m_i}} > U'. \tag{9}$$

The result in this case is:

$$X = \left\lceil \frac{\log\big(q(L_i - 1) - U'(1 - e^{-m_i})\big) - b_i}{m_i} \right\rceil$$

Finally, when $m_i = 0$, then $q(\cdot)$ is constant over the piece, and Condition (7) simplifies to:

$$q(L_i)(X - L_i + 1) > U',$$

The smallest integer X for which this holds is

$$X = \left\lceil \frac{U'}{q(L_i)} + L_i - 1 \right\rceil.$$

Algorithm 4: Sampling From the Upper Bound

```
Input  : Piecewise-linear function $h(k)$ defined by pieces $h_i$ with parameters $(m_i, b_i, L_i, U_i)$
Output: Random number $X$ from the distribution with mass function $q(k) = \exp h(k)$
// Compute mass of each piece
$C \leftarrow \max\Big(\max_{i:m_i<0}(m_iL_i + b_i), \max_{i:m_i>0}(m_iU_i + b_i)\Big)$;
for $i = 1$ to $n$ do
    $b'_i \leftarrow b_i - C$;
    $Z_i \leftarrow \begin{cases} \dfrac{\exp(m_iL_i + b'_i) - \exp(m_i(U_i+1) + b'_i)}{1 - \exp(m_i)} & m_i < 0, \\ \exp(m_iL_i + b'_i)(U_i - L_i + 1) & m_i = 0, \\ \dfrac{\exp(m_iU_i + b'_i) - \exp(m_i(L_i-1) + b'_i)}{1 - \exp(-m_i)} & m_i > 0. \end{cases}$
end
$Z \leftarrow \sum_{i=1}^n Z_i$;
$U \leftarrow \text{Uniform}([0,1])$;
$i \leftarrow$ smallest integer such that $Z_1 + \ldots + Z_i > UZ$;
$U' \leftarrow UZ - (Z_1 + \ldots + Z_i)$;
$X \leftarrow \begin{cases} \left\lceil \dfrac{\log\big(\exp(mL_i + b'_i) - U'(1 - e^{m_i})\big) - b'_i}{m_i} - 1 \right\rceil & m_i < 0, \\ \left\lceil \dfrac{U'}{\exp(mL_i + b'_i)} + L_i - 1 \right\rceil & m_i = 0, \\ \left\lceil \dfrac{\log\big(\exp(m(L_i-1) + b'_i) - U'(1 - e^{-m_i})\big) - b'_i}{m_i} \right\rceil & m_i > 0. \end{cases}$
```

Avoiding Overflow. Finally, to avoid numerical overflow and underflow, we follow the standard practice of subtracting the constant value C from the piecewise-linear function h in each of the above computations to ensure that the maximum value being exponentiated is not too big. The correct value to subtract is:

$$C = \max\Big(\max_{i:m_i<0}(m_iL_i + b_i), \max_{i:m_i>0}(m_iU_i + b_i)\Big).$$

This amounts to scaling $\exp h$ by a constant, which does not change the probability distribution it defines.

The complete sampling procedure is showing in Algorithm 4.

References

W.R. Gilks and P. Wild. Adaptive Rejection sampling for Gibbs Sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):337–348, 1992. ISSN 0035-9254. URL `http://www.jstor.org/stable/2347565`.

L. Devroye. *Non-uniform random variate generation*, volume 4. Springer-Verlag New York, 1986.

Luc Devroye. A note on generating random variables with log-concave densities. *Statistics & Probability Letters*, 82(5):1035 – 1039, 2012. ISSN 0167-7152. doi: 10.1016/j.spl.2012.01.022. URL `http://www.sciencedirect.com/science/article/pii/S0167715212000326`.

L. Devroye. A simple algorithm for generating random variates with a log-concave density. *Computing*, 33(3-4):247–257, September 1984. ISSN 0010-485X. doi: 10.1007/BF02242271. URL `http://www.springerlink.com/index/10.1007/BF02242271`.

L. Devroye. A simple generator for discrete log-concave distributions. *Computing*, 39(1):87–91, 1987.

W. Hörmann, J. Leydold, and G. Derflinger. *Automatic nonuniform random variate generation.* Springer Verlag, 2004.