8 Stochastic Sampling

8.1 Continuous Probabilities

When defining discrete probabilities, we took care to separate the universe Ω (the set of all outcomes) from the event space $\mathcal{E} = 2^\Omega$ (the class of all sets of outcomes), and assigned probabilities to elements of $\mathcal{E}$ rather than of Ω. We then defined a random variable X as a function from Ω into a subset $\mathcal{X}$ of the integers, and a probability distribution $p_X(x)$ as the following function from $\mathcal{X}$ into the real interval $[0, 1]$:

$$p(x) = P[X^{-1}(x)] \quad \text{for all} \quad x \in \mathcal{X} \ .$$

For any given x in $\mathcal{X}$ the expression $X^{-1}(x)$ denotes a set of outcomes, that is, an element of $\mathcal{E}$. This is appropriate, since probabilities are defined on $\mathcal{E}$.

In the context of discrete variables, this construction seems a bit roundabout: Why not just define a probability for each of the outcomes ω in the universe Ω? The reason is that it extends to the case of continuous variables, as shown next.

Suppose that the universe Ω is not discrete. For instance, a point $\omega \in \Omega$ could represent the loudness of a sound, for which discrete outcomes are not a natural description. The novelty in this situation is that the probability of any singleton event $\{\omega\}$ is zero (it is extremely unlikely that an arbitrary level of loudness will be met exactly), so assigning probabilities to outcomes would be meaningless. Instead, we define a random variable X on Ω (just as for discrete outcomes) to transform outcomes into real numbers (say the voltage produced by a microphone at a fixed distance from the sound source). Then, we assign probabilities to events of the form $X(\omega) \leq x$ for any $x \in \mathbb{R}$:

$$F_X(x) = P(\{\omega \ : \ X(\omega) \leq x\}) \ .$$

The function $F_X(x)$ is called the *(cumulative) probability distribution* of X. Since for $x_1 < x_2$ we have

$$\{\omega \ : \ X(\omega) \leq x_2\} = \{\omega \ : \ X(\omega) \leq x_1\} \ \cup \ \{\omega \ : \ x_1 < X(\omega) \leq x_2\}$$

and the two events on the right-hand side are disjoint, the third axiom $\mathcal{P}_3$ of probability implies that

$$P(\{\omega \ : \ X(\omega) \leq x_2\}) = P(\{\omega \ : \ X(\omega) \leq x_1\}) \ + \ P(\{\omega \ : \ x_1 < X(\omega) \leq x_2\}) \ ,$$

that is,

$$F_X(x_2) = F_X(x_1) + P(\{\omega \ : \ x_1 < X(\omega) \leq x_2\}) \geq F_X(x_1)$$

because probabilities are nonnegative (first axiom). So

$$x_1 < x_2 \Rightarrow F_X(x_1) \leq F_X(x_2) \ .$$

Also,

$$F(-\infty) = P(\{\omega \ : \ X(\omega) \leq -\infty\}) = P(\emptyset) = 0$$

and

$$F(\infty) = P(\{\omega \ : \ X(\omega) \leq \infty\}) = P(\Omega) = 1$$

from the second axiom. Thus, the distribution function of any random variable is a monotonically nondecreasing function with values between 0 and 1.

Its derivative,

$$f_X(x) = \frac{dF(x)}{dx}$$

is called the *probability density* of X. Note that $F_X(x)$ is a probability for each x, but $f_X(x)$ is not. Since

$$F(x) = \int_{-\infty}^{x} f(\xi)\, d\xi \ ,$$

we can write

$$F_X(x_2) = F_X(x_1) + P(\{\omega \ : \ x_1 < X(\omega) \leq x_2\}) \geq F_X(x_1) = \int_{x_1}^{x_2} f(\xi)\, d\xi \ .$$

This provides the following interpretation of $f_X(x)$:

> The integral of the probability density function $f_X(x)$ over an interval $x_1 < x \leq x_2$ is equal to the probability that the random variable X takes values in that interval.

Alternatively, if $x_2 - x_1$ is positive, but so small that $f_X(x)$ is approximately constant for $x_1 < x \leq x_2$, then we can say the following:

> The probability density $f_X(x)$ is a value such that the product $f_X(x)\, dx$ for an infinitesimally small, positive dx equals the probability that X is in an interval of width dx around x.

From this we also see that $f_X(x)$ is always nonnegative.

Just as for discrete probabilities, one can then define moments through the notion of *mean*, or expectation:

$$m_X = E[X] = \int_{-\infty}^{\infty} x\, f(x)\, dx$$

and more generally

$$m_{g(x)} = E[g(X)] = \int_{-\infty}^{\infty} g(x)\, f(x)\, dx$$

for any function $g \ : \ \mathbb{R} \to \mathbb{R}$. The *variance* is an important special case:

$$\sigma_X^2 = E[(X - m_X)^2] = \int_{-\infty}^{\infty} (x - m_X)^2\, f(x)\, dx \ .$$

8.2 Sampling and Integration

Many situations in modeling require finding statistical summaries of quantities in the form of averages, or to draw out of a distribution. For instance, studying the properties of certain liquids may require simulating plausible positions of a large number of particles inside a container. This in turn amounts to drawing a sample

$$\mathbf{x} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]^T$$

of positions in some volume of space (so $\mathbf{x}_n$ is a vector with three Cartesian coordinates, one vector per particle, and $\mathbf{x}$ lists $3N$ coordinates) out of a probability density that describes plausible configurations, derived from an understanding of the relevant physics. In the particular instance, the probability density can be shown to be of the following, complicated form[26]

$$f(\mathbf{x}) = \frac{e^{-\frac{1}{k_b T}\sum_{m<n=1}^{N}\Phi(\|\mathbf{x}_m - \mathbf{x}_n\|)}}{\int_{\mathbb{R}^{3N}} e^{-\frac{1}{k_b T}\sum_{m<n=1}^{N}\Phi(\|\mathbf{x}_m - \mathbf{x}_n\|)}\, d\mathbf{x}}$$

where

$$\Phi(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right].$$

In these expressions, T is the temperature of the liquid (in Kelvin), $k_b \approx 1.38 \times 10^{-23}$ is the Boltzmann constant, and ϵ and σ are constants that depend on the type of liquid. This probability distribution is called the *Boltzmann distribution* for the liquid in question.

The expression for the probability density $f(\mathbf{x})$ is complicated to evaluate, because the integral at the denominator is essentially impossible to compute analytically. Second, even if it were possible to evaluate $f(\mathbf{x})$, sampling from it would require finding values of the vector $\mathbf{x}$ for which $f(\mathbf{x})$ is large (that is, likely configurations for the gas). How can we find those values without searching all of $\mathbb{R}^{3N}$? This is a space of astronomical dimensions!

In general, the following two problems are of interest in this type of problems:

P_1**:** Draw from a distribution $f(\mathbf{x})$.

P_2**:** Evaluate an expectation,

$$E[g(\mathbf{X})] = \int_D g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}$$

where $f(\mathbf{x})$ is a probability density and D is the domain of integration.

These problems are closely related to each other, in two different ways: First, as we just saw, the solution of P_1 might require computing an integral, that is, solving P_2 (let $f(\mathbf{x})$ be the Boltzmann distribution, and let $g(\mathbf{x})$ be the denominator of the same). Second we have seen (for discrete variables, but the same result holds for continuous one) that a statistical expectation can be estimated by a sample mean:

$$E[g(\mathbf{X})] \approx \frac{1}{M}\sum_{m=1}^{M} g(\mathbf{x}^{(m)})$$

[26]The main point here is that the form is complicated. No explanation for this expression is given here.

where $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$ are M samples drawn out of $f(\mathbf{x})$. So computing P_2 could be achieved by drawing M samples out of $f(\mathbf{x})$ (that is, solving P_1), computing $g(\mathbf{x}^{(m)})$ for each of them, and averaging the values thus obtained.

In the following, we discuss methods for solving problems P_1 and P_2 in many dimensions. As it turns out, these methods can be very inefficient (that is, very expensive to compute), so we also look at a more efficient method, the Metropolis-Hastings algorithm, thereafter.

8.3 Rejection Sampling

We have discussed a sampling method for discrete probability distributions $p(x)$ of a single variable: Find the cumulative distribution function

$$c(x) = \sum_{\xi \leq x} p(\xi) \ ,$$

then draw a random number r uniformly from $[0, 1]$, and let the sample be the smallest x for which

$$c(x) \geq r \ .$$

This method could be extended to continuous densities $f(x)$ by replacing $c(x)$ with the probability distribution function $F(x)$. This is a continuous[27] function with values in $[0, 1]$, so we now just need to solve the following for x:

$$F(x) = r \ .$$

This method, however, cannot extend to multiple dimensions, since the solution to the last equation then fails to be unique. For instance, if $\mathbf{x} \in \mathbb{R}^2$, there is a whole curve of values for which the equation is satisfied (for any given $r \in [0, 1]$).

Here, on the other hand, is a method that extends in principle to any number of dimensions.

Rejection Sampling
Let $h(\mathbf{x})$ be any distribution for which there exists a constant c such that

$$f(\mathbf{x}) \leq ch(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \ ,$$

and from which samples can be drawn. Draw a point $\mathbf{x}$ randomly out of $h(\mathbf{x})$. Then, draw a number u uniformly between 0 and $ch(\mathbf{x})$. Accept $\mathbf{x}$ if

$$u \leq f(\mathbf{x}) \ .$$

Otherwise, reject (*i.e.*, discard) $\mathbf{x}$, and repeat.

[27] Assuming no discrete component to the distribution.

This method is illustrated most easily for a one-dimensional density $f(x)$ that is zero outside the interval $a \leq x \leq b$. Then, an easy choice for $h(x)$ is the uniform distribution over $[a, b]$:

$$h(x) = \frac{1}{b-a} \quad \text{for} \quad a \leq x \leq b \quad \text{and zero elsewhere,}$$

and

$$c = (b-a) \max_{x \in [a,b]} f(x) \ .$$

In this case, x is drawn uniformly in $[a, b]$, and (x, u) before rejection is a point distributed uniformly in the rectangle with base $[a, b]$ and height c. Therefore, (x, u) after rejection is a point distributed uniformly in the area between the x axis and the graph of $f(x)$, so that x is distributed according to $f(x)$.

Note that this reasoning holds even if $f(x)$ is multiplied by a positive constant: what counts is that the random abscissas x are drawn with a frequency that is proportional to $f(x)$. This is important, since we saw earlier that a probability density is often known only up to a multiplicative constant: rejection sampling works in that case as well, and without any modification.

> Both drawings (of x and of u) in the rejection sampling method can be achieved easily in Matlab by using properly scaled and shifted versions of `rand`:
>
> $$\texttt{x} = \texttt{a} + (\texttt{b} - \texttt{a}) * \texttt{rand}; \quad \text{and} \quad \texttt{u} = \texttt{c} * \texttt{rand};$$
>
> If on the other hand $f(x)$ is defined over an unbounded interval, $h(x)$ can be chosen to be the Gaussian density, for which the Matlab function `randn` is available.

The same reasoning used for one-dimensional distributions can be used in multiple dimensions, so that the rejection sampling method extends in principle to any number of dimensions for the domain of $f(\mathbf{x})$. In particular, the uniform distribution on an axis-aligned box is just the product of uniform distributions on each of the axes:

$$U_n([a_1, b_1] \times \ldots \times [a_n, b_n]) = U_1([a_1, b_1]) \cdot \ldots \cdot U_1([a_n, b_n])$$

and a similar property, but on all of $\mathbb{R}^n$ holds for Gaussians with a diagonal covariance matrix:

$$G_n([m_1, \ldots, m_n], \begin{bmatrix} \sigma_1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \sigma_n \end{bmatrix}) = G_1(m_1, \sigma_1) \cdot \ldots \cdot G_1(m_n, \sigma_n) \ .$$

> So to draw a single sample out of a standard uniform or Gaussian distribution in n dimensions you would type
>
> $$\texttt{rand(n, 1)} \quad \text{or} \quad \texttt{randn(n, 1)}$$
>
> and to draw M samples at once you would type
>
> $$\texttt{rand(n, M)} \quad \text{or} \quad \texttt{randn(n, M)} \ .$$

A drawback of the rejection sampling method is that it can become inefficient if there is a large volume between $f(\mathbf{x})$ and its upper bound $ch(\mathbf{x})$, since many pairs $(\mathbf{x}, u)$ may have to be generated for each pair that is accepted. This problem becomes more severe with increasing dimensionality, because the volume between the two functions has a greater likelihood of becoming bigger in multiple dimensions: We do not know a priori where the mass of $f(\mathbf{x})$ is concentrated, and so the function $ch(\mathbf{x})$ must be broad enough to cover any possible candidate volume in $\mathbb{R}^n$. As n increases, the ratio of the volume "under" $f(\mathbf{x})$ (one, if $f(\mathbf{x})$ is normalized) to the volume "under" $ch(\mathbf{x})$ often becomes vanishingly small, and efficiency drops to very low values as a consequence.

This issue will be addressed later on, when describing the Metropolis-Hastings algorithm, which is also a sampling method based on a rejection mechanism. For now we consider the closely related issue of integration.

8.4 Monte Carlo Integration

Now that we have a method for drawing out of a probability density $f(\mathbf{x})$, computing integrals is simply a matter of approximating a probabilistic expectation with a sample mean:

Monte Carlo Integration: To compute the expectation

$$E[g(\mathbf{X})] \;=\; \int_D g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}$$

where D is the integration domain, draw M samples

$$\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$$

out of $f(\mathbf{x})$ and let

$$E[g(\mathbf{X})] \approx \frac{1}{M} \sum_{m=1}^{M} g(\mathbf{x}^{(m)}) \;.$$

To compute an arbitrary definite integral

$$I \;=\; \int_D a(\mathbf{x})\, d\mathbf{x} \;,$$

let $f(\mathbf{x})$ be any probability density that is nonzero over the entire domain of integration, and compute

$$I = E\left[\frac{a(\mathbf{x})}{f(\mathbf{x})}\right]$$

by the method above (that is, with $g(\mathbf{x}) = a(\mathbf{x})/f(\mathbf{x})$).

8.5 The Metropolis Algorithm

As mentioned earlier, the rejection method for drawing out of a given density $f(\mathbf{x})$ can be inefficient, and more frequently so if $\mathbf{x}$ has more dimensions. This problem is addressed by the Metropolis algorithm described in this Section. Note that the Monte Carlo integration method given above is independent of how one draws from the given density $f(\mathbf{x})$, so if we fix sampling we also fix integration as a consequence.

Rather than drawing independent abscissas $\mathbf{x}$, the Metropolis algorithm draws highly correlated ones: it starts at a random point $\mathbf{x}^{(1)}$, and then proposes a "move," by adding a random, zero-mean vector $\mathbf{r}$ to $\mathbf{x}^{(1)}$:

$$\mathbf{x}' = \mathbf{x}^{(1)} + \mathbf{r} \ .$$

Note that the new position is called $\mathbf{x}'$ rather than $\mathbf{x}^{(2)}$, because this is only a tentative proposal, which might be rejected. In principle, it does not matter how, that is, according to what density, $\mathbf{r}$ is generated. This density is called the *proposal density*.

Should the algorithm move to $\mathbf{x}'$ or not? If $f(\mathbf{x}') \geq f(\mathbf{x}^{(1)})$, the answer is yes, because the next sample is at least as likely as the current one. If $f(\mathbf{x}') < f(\mathbf{x}^{(1)})$, the answer is less obvious: the algorithm cannot reject the move every time, because then it would get stuck at the most likely position, a maximum of $f(\mathbf{x})$. However, it should discourage more strongly moves to much less likely positions. To this end, the algorithm draws a uniformly random number u between 0 and 1. If $f(\mathbf{x}') \geq u f(\mathbf{x}^{(1)})$, then the move is accepted, and the new sample $\mathbf{x}^{(2)}$ is set to $\mathbf{x}'$. Otherwise, the move is rejected, and the new sample $\mathbf{x}^{(2)}$ is set to be equal to the old sample $\mathbf{x}^{(1)}$. Thus, moves to less likely positions are discouraged (by a multiplicative factor $u \leq 1$), but not disallowed altogether. Although we do not prove this here, this policy results in drawing from the desired density.

A more formal description of the method is given in the box on page 106.

Note that if the proposed position is more likely than the old one,

$$f(\mathbf{x}') \geq f(\mathbf{x}^{(m-1)}) \ ,$$

then *a fortiori*

$$f(\mathbf{x}') \geq u f(\mathbf{x}^{(m-1)}) \ ,$$

so the proposal is accepted in any case. Because of this, it may be more efficient to first check the first condition above, and generate u (a more expensive operation) only if it is violated, that is only if

$$f(\mathbf{x}') < f(\mathbf{x}^{(m-1)}) \ .$$

In this case, the probability that

$$f(\mathbf{x}') \geq u f(\mathbf{x}^{(m-1)})$$

for positive $f(\mathbf{x}^{(m-1)})$ is

$$P(f(\mathbf{x}') \geq u f(\mathbf{x}^{(m-1)})) = P(u \leq \frac{f(\mathbf{x}')}{f(\mathbf{x}^{(m-1)})}) = \frac{f(\mathbf{x}')}{f(\mathbf{x}^{(m-1)})}$$

Metropolis Algorithm: Given a nonnegative, nonzero function

$$f \ : \ \mathbb{R}^n \to \mathbb{R}^+ \ ,$$

an initial sample $\mathbf{x}^{(1)}$, and a zero-mean *proposal density* $h(\mathbf{x})$ from which it is known how to draw, generate consecutive samples $x^{(2)}, \ldots, x^{(M)}$ out of the density

$$\frac{f(\mathbf{x})}{\int_{\mathbb{R}^n} f(\mathbf{x})\, d\mathbf{x}}$$

as follows:

for $m = 2$ to M
  $\mathbf{r} \ \leftarrow$ sample out of $h(\mathbf{x})$
  $\mathbf{x}' \ \leftarrow \ \mathbf{x}^{(m-1)} + \mathbf{r}$
  $u \ \leftarrow$ sample out of a uniform distribution on $[0, 1]$
  if $f(\mathbf{x}') \geq u f(\mathbf{x}^{(m-1)})$
    $\mathbf{x}^{(m)} \ \leftarrow \ \mathbf{x}'$
  else
    $\mathbf{x}^{(m)} \ \leftarrow \ \mathbf{x}^{(m-1)}$
  end
end

because u is uniformly distributed over $[0, 1]$. Thus, if $f(\mathbf{x}^{(m-1)}) > 0$, a more likely proposal is always accepted, and a less likely proposal is accepted with probability equal to the *likelihood ratio*

$$\rho = \frac{f(\mathbf{x}')}{f(\mathbf{x}^{(m-1)})} \ .$$

If $f(\mathbf{x}^{(m-1)}) = 0$, the proposal is always accepted, because

$$f(\mathbf{x}') \geq u \cdot 0 = 0$$

in any case.

A typical choice for the proposal distribution $h(\mathbf{x})$ is a Gaussian with zero mean and isotropic standard deviation σ.

The path traversed by the sequence $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$ is called a *random walk*. The first steps of the walk depend on the choice of starting point $\mathbf{x}^{(1)}$, which is arbitrary. Because of this, the first several steps, called the *burn-in* sequence, are discarded, to allow the walk to "forget" its starting point or, in technical jargon, to *mix*.

If σ is too small, the walk moves very slowly, and burn-in is long. If σ is too large, the random samples are more likely to hit regions where $f(\mathbf{x})$ is small, so very low acceptance rates will ensue and the walk moves again very slowly. A rule of thumb is to tune σ so as to achieve acceptance rates around $1/2$.

Note that what is accepted or rejected by the Metropolis algorithm is the proposed step. If this is rejected, the previous sample $x^{(m-1)}$ is copied as the new sample $x^{(m)}$. So strictly speaking the efficiency of the Metropolis algorithm is 100 percent: although the *new* sample may be rejected, a sample (possibly equal to the previous one) is obtained at every step. In addition, once the algorithm has found a volume where $f(\mathbf{x})$ is large, it tends to linger there (because it is after all, drawing from $f(\mathbf{x})$!), producing new, nearby samples at high rate. Because of this, consecutive samples are highly correlated. This is what "nearby" implies: if we know where the current sample is, we have a good idea for where the next one is going to be. Correlation is an undesired property for a sampling algorithm, since statistical analysis usually assumes independent (and therefore uncorrelated) samples. However, this property is also the reason why the Metropolis algorithm is much more efficient than the standard rejection sampling method: "lingering" (in parts of the space where $f(\mathbf{x})$ is high) implies that consecutive samples are highly correlated. Thus, efficiency comes at a price. Various tradeoffs can be achieved (by tuning σ and by other means), but there is no free lunch.

8.6 Rejection Versus Metropolis in High Dimensions

The Metropolis algorithm for sampling was motivated by the observation that rejection sampling tends to have low efficiency when the dimension n of the space is large: there is a large volume between $ch(\mathbf{x})$ (the proposal density $h(\mathbf{x})$, scaled so that $f(\mathbf{x}) \leq ch(\mathbf{x})$ everywhere) and $f(\mathbf{x})$, so many of the samples are rejected. This volume is mostly where $f(\mathbf{x})$ is small, as opposed to where $ch(\mathbf{x})$ is large, because normalization implies that $f(\mathbf{x})$ is small almost everywhere.

In contrast, the Metropolis sampler tends to move towards higher values of f (because of the bias inherent in the rejection decisions), and to linger where values of f are high (for the same reason). These are regions of comparatively high efficiency, so it stands to reason that the Metropolis algorithm is more efficient than the rejection sampler.

To see this more quantitatively, a classical case study is to see what happens when both the "unknown" density $f(\mathbf{x})$ and the proposal density $h(\mathbf{x})$ are isotropic[28] Gaussian densities. This is because in this case the math is easy to work out, and the volumes under the various curves can be computed analytically. Of course this is only for the purpose of performance analysis: We would never use one Gaussian to sample from another one, if we knew the parameters of the latter!

In n dimensions, the expression of the isotropic Gaussian density with zero mean and standard deviation σ in all directions is

$$g_{n,\sigma}(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2}\left(\frac{\|\mathbf{x}\|}{\sigma}\right)^2}, \tag{46}$$

[28] Isotropic here means that the spread of the Gaussian is the same in all directions of space.

where

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \ldots + x_n^2}$$

is the norm of $\mathbf{x}$ (that is, its Euclidean distance from the origin). Since densities are normalized,

$$\int_{\mathbb{R}^n} g_{n,\sigma}(\mathbf{x})\, d\mathbf{x} = 1 \ ,$$

equation (46) implies that if we let

$$\hat{g}_{n,\sigma}(\mathbf{x}) = e^{-\frac{1}{2}\left(\frac{\|\mathbf{x}\|}{\sigma}\right)^2}$$

be the un-normalized Gaussian with maximum value

$$\hat{g}_{n,\sigma}(\mathbf{0}) = 1 \ ,$$

then

$$\int_{\mathbb{R}^n} \hat{g}_{n,\sigma}(\mathbf{x})\, d\mathbf{x} = (2\pi\sigma^2)^{n/2} \ .$$

So assume that we try to sample out of

$$f(\mathbf{x}) = g_{n,1}(\mathbf{x})$$

with a proposal distribution

$$h(\mathbf{x}) = g_{n,1+\epsilon}(\mathbf{x}) \ ,$$

where $\epsilon > 0$ is a small number. In other words, we try to sample from a unit-variance Gaussian, using a slightly wider Gaussian as the proposal distribution. We know that common constants do not matter, so we can use

$$\hat{f}(\mathbf{x}) = \hat{g}_{n,1}(\mathbf{x}) \quad \text{and} \quad \hat{h}(\mathbf{x}) = \hat{g}_{n,1+\epsilon}(\mathbf{x})$$

instead of $f(\mathbf{x})$ and $h(\mathbf{x})$. This simplifies reasoning, because the maximum value of these functions is the same:

$$\hat{f}(\mathbf{0}) = \hat{h}(\mathbf{0}) = 1$$

so we can let $c = 1$ in the rejection sampling algorithm. Since $\hat{h}(\mathbf{x})$ is slightly wider than $\hat{f}(\mathbf{x})$, it dominates everywhere:

$$\hat{f}(\mathbf{x}) \leq \hat{h}(\mathbf{x}) \ ,$$

as needed for rejection sampling.

The volume under these functions is

$$\int_{\mathbb{R}^n} \hat{f}(\mathbf{x})\, d\mathbf{x} = \int_{\mathbb{R}^n} \hat{g}_{n,1}(\mathbf{x})\, d\mathbf{x} = (2\pi)^{n/2}$$

and

$$\int_{\mathbb{R}^n} \hat{h}(\mathbf{x})\, d\mathbf{x} = \int_{\mathbb{R}^n} \hat{g}_{n,1+\epsilon}(\mathbf{x})\, d\mathbf{x} = (2\pi(1+\epsilon)^2)^{n/2} \ .$$

The efficiency of rejection sampling is the ratio of these volumes:

$$\eta(n,\epsilon) = \frac{\int_{\mathbb{R}^n} \hat{f}(\mathbf{x})\,d\mathbf{x}}{\int_{\mathbb{R}^n} \hat{h}(\mathbf{x})\,d\mathbf{x}} = \frac{(2\pi)^{n/2}}{(2\pi(1+\epsilon)^2)^{n/2}} = \frac{1}{(1+\epsilon)^n}\ ,$$

which decays exponentially with the dimension n. Figure 21 shows the decay for $\epsilon = 0.01$. This is so fast that a logarithmic scale is used for the vertical axis. As a numerical example,

$$\eta(1000, 0.01) \approx 1/21000\ ,$$

so even with only a slightly wider proposal distribution, efficiency drops to 1 acceptance every 21,000 attempts!

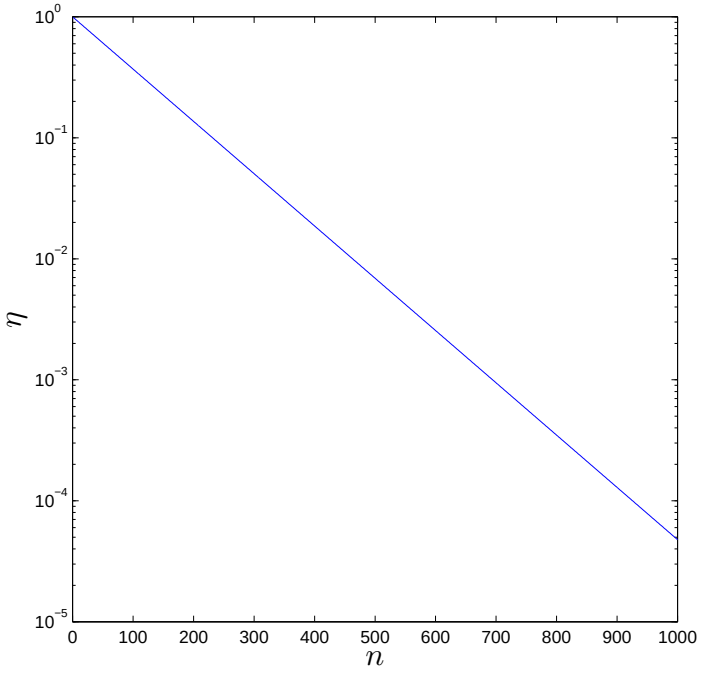

Figure 21: The efficiency $\eta(n, 0.01)$ of rejection sampling for a unit-variance Gaussian, and with a Gaussian with variance 1.01, plotted as a function of the dimension n of the underlying space.

The situation with the Metropolis algorithm is subtler, because the time needed to "cover" a Gaussian depends on the shape of the latter. Consider for instance sampling out of a two-dimensional Gaussian that is more elongated in the x_1 direction than in the x_2 direction. If the proposal distribution is $g_{n,\epsilon}(\mathbf{x})$ (a small, isotropic Gaussian), then it will take comparatively more steps to cover the x_1 direction than the x_2 direction. Figure 22 shows accepted and rejected samples for three different values of ϵ.

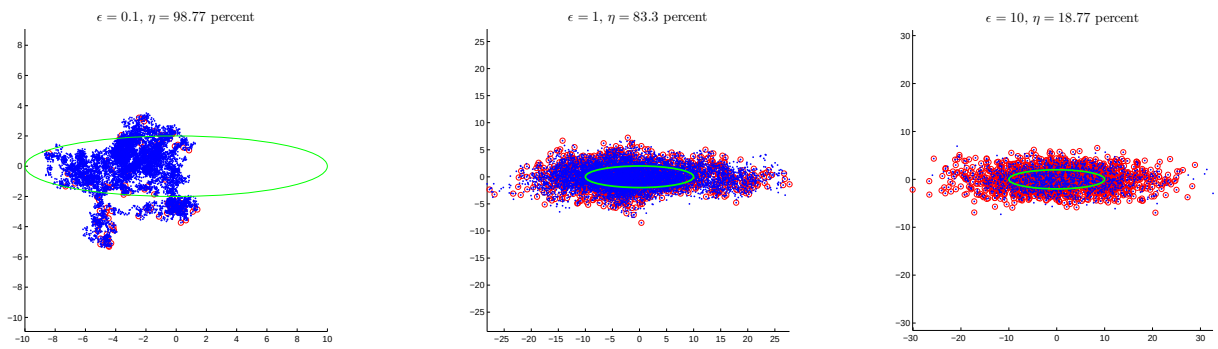

Figure 22: 10,000 samples generated by the Metropolis algorithm with three different values of ϵ. The green ellipse contains about 95 percent of the mass of the Gaussian $f(\mathbf{x})$.

Note that if $\epsilon << \sigma_{\max}$ (plot on the left), then the random walk generated by the Metropolis algorithm meanders around in tiny steps, and a very long walk is needed to cover the density $f(\mathbf{x})$. However, efficiency is high: Once the walk is on the relatively flat tails of the Gaussian, the ratio $f(\mathbf{x}')/f(\mathbf{x}^{(m-1)})$ between the values of f at the new proposed location $\mathbf{x}'$ and at the previous position $\mathbf{x}^{(m-1)}$ is everywhere close to 1, so that the acceptance check

$$f(\mathbf{x}') \geq u f(\mathbf{x}^{(m-1)})$$

is very likely to succeed (recall that u is random between 0 and 1).

On the other hand, if ϵ is of the order of $\sigma_{\max}$ or greater (plot on the right in Figure 22), each step goes to an entirely different position, relative to the extent of $f(\mathbf{x})$, so the samples are virtually uncorrelated, and the Metropolis algorithm becomes similar to the rejection method. The efficiency is consequently low (and more so in many dimensions). Either way (ϵ too small or too large), it takes many samples to visit a significant portion of the density $f(\mathbf{x})$.

Intermediate situations (center plot in Figure 22) are best: steps are significant, and yet efficiency is fairly high. Elements of a theory for choosing good step sizes in more general situations exist, but are beyond the scope of these notes.