Week 13: Wednesday, Apr 27

Logistics

- Project 3 is due Friday, May 6.
- Final exam is Friday, May 13.

Problem du jour

Suppose that each homework problem I attempt is an independent Bernoulli random variable, with probability of success equal to 0.8. I'm in a class with seven homeworks of five points each, and the lowest of these grades is dropped. What is my expected average score for the semester?

Answer: We can do this in closed form, but it's easier to set up the calculation using brute force Monte Carlo:

```
n = 10000;
for k = 1:n
  sample   = sort( sum(double(rand(7,5) <= 0.8),2) );
  score(k) = sum( sample(2:end) )/6;
end
mean(score);
fprintf('%f (%.1e)\n', mean(score), std(score)/sqrt(n));
```

The answer is that the average score will be a bit over 4.2 (84%).

I would usually do a Monte Carlo computation in a case like this even if I worked out the exact solution, just so that I would have a sanity check on my algebra.

Random number generation

At the end of the last lecture, we were just starting to touch on the problem of drawing (pseudo)random samples from different distributions. Recall that we are going to assume that we have a reasonably good method for producing samples from the uniform distribution on $(0, 1)$. The question in how we draw samples from other distributions. I know a handful of tricks for deriving new samplers; let's investigate them by example.

Bernoulli random variables

A Bernoulli random variable generates 1 (success) with probability p and 0 (failure) with probability $1 - p$. In the problem du jour, we implicitly assumed that each question was a Bernoulli trial with $p = 0.8$. Generating a Bernoulli trial from a uniform sampler is relatively simple; in MATLAB, we might write

```
function result = bernoulli(p)

  U = rand(1); % Generate a sample in Unif(0,1)
  if U < p
    result = 1;
  else
    result = 0;
  end
```

Note that $P\{U < p\} = \int_0^p 1 \, du = p$, so this sampler certainly has the right properties. Also notice that in my pseudocode for the problem du jour, I generated 35 Bernoulli trials simultaneously using one call to `rand`. The random number generator in MATLAB has a fair amount of overhead per call, so you really want to generate random variables in reasonably large blocks if you can.

The project 3 code uses Bernoulli trials in the Russian roulette algorithm for deciding when to terminate a trajectory (or group of trajectories).

Exponential random variables

An exponential random variable with rate parameter λ has the density function

$$f(x; \lambda) = \lambda e^{-\lambda x}, \quad x \geq 0$$

and the cumulative distribution function

$$F(x; \lambda) = 1 - e^{-\lambda x}.$$

Now, suppose that for a uniform sample U we generate X to satisfy $F(X; \lambda) = U$, i.e.

$$X = -\frac{1}{\lambda} \log(1 - U).$$

Then

$$P\{X \le x\} = P\{F(X;\lambda) \le F(x;\lambda)\} = P\{U \le F(x;\lambda)\} = F(x;\lambda).$$

This inverse transformation trick works whenever we have a simple way to compute a cumulative distribution function.

The step sizes in the Monte Carlo simulator in project 3 are exponential random variables drawn using exactly this trick.

Sampling from an empirical distribution

Suppose we have a histogram of results from some large number of real-world experiments. If the outcomes of the experiments are integers in the range from 1 to m, we can define a probability mass function where $p(j)$ is the fraction of the experiments that had outcome j. There is a corresponding cumulative distribution function $F(j) = \sum_{i=1}^{j} p(j)$ that goes from $F(0) = 0$ to $F(m) = 1$. To draw a sample from this distribution, we would again use the inverse transformation trick: draw U uniform between 0 and 1, then choose the smallest j such that $F(j) > U$.

Sampling from the unit disk

Suppose we want to draw (X, Y) uniformly at random from the interior of the unit circle. One way to do this is with polar coordinates: if U_1 and U_2 are uniform on $(0, 1)$, we can generate $\Theta = 2\pi U_1$ and $R = \sqrt{U_2}$ (the cdf for R should be $F_R(r) = r^2$ on $[0, 1)$, so we can use the inverse transformation trick from above). Then we could compute $(X, Y) = R(\sin\Theta, \cos\Theta)$. But suppose we didn't know this, or suppose that we're thinking of the disk as a proxy for some more complicated set sitting inside the unit square. What other tactics could we use?

One simple idea is *rejection sampling*. The basic idea is

- Draw a sample from an easy distribution g. In this case, we might use the uniform distribution on $[-1, 1]^2$ (i.e. $g(x, y) = 1/4$ on $[-1, 1]^2$ and zero elsewhere).
- Accept the sample with probability that is a function of the sample values. In this case, we have

$$p(x, y) = \begin{cases} 1, & x^2 + y^2 < 1 \\ 0, & \text{otherwise.} \end{cases}$$

In this case, we accept with probability one if $X^2 + Y^2 < 1$ and with probability zero otherwise.

We then keep repeating until acceptance. The probability density associated with the accepted values is then

$$f(x, y) = \frac{1}{Z} g(x, y) p(x, y)$$

where Z is some normalization constant chosen so that the acceptance probability is one. In our case, this gives us a density that is a nonzero constant on the circle and zero elsewhere, which is what we wanted.

A more geometric way of seeing rejection sampling is that we fit some shape that completely surrounds the graph of our density function (in this case, that shape is a three-dimensional box). We then draw uniformly at random from within that shape, and discard the samples that do not fall under the graph of the density function. The probability that we succeed in any given trial is equal to the fraction of the area inside the shape that lies underneath the graph of the density function.

Distribution with an exponential tail

Let's look at another example of rejection sampling. Suppose I wanted to sample from $f(x) = C^{-1} g(x) e^{-x}$ on $[0, \infty)$, where C is some (possibly unknown) normalization constant and $0 < g(x) < G$. Then I could compute samples from f using the following procedure:

```
p = 0;                  % p = probability of accepting current X
while (rand(1) > p)     % while we fail to accept
  X = -log(rand(1));    %   X is an exponential random variable
  p = g(X)/G;           %   p is proportional to g(X)
end
```

The probability of success in this problem is the ratio of the area under the histogram for f to the area under Ge^{-x}, or $1/G$. The expected number of rounds until success is therefore G.