AMTH142

Lecture 14

Random Numbers

This and the next couple of lectures will be on random numbers and simulation.

Contents

1 Review of Probability 2
   1.1 Definitions ........................................... 2
   1.2 Uniform Density ........................................ 3
   1.3 Normal Density ......................................... 3

2 Random Number Generators 3
   2.1 Algorithms ............................................. 4
      2.1.1 Linear Congruential Generators ................. 4
      2.1.2 Fibonacci Generators ............................ 6
      2.1.3 Practical Generators ............................. 6

3 Scilab 6
1 Review of Probability

This is just a reminder of few facts about continuous probability distributions covered in Math102.

1.1 Definitions

Continuous probability distributions are described by probability densities, i.e. real valued functions $\rho(x)$ with the properties:

1. $\rho(x) \geq 0$.

2. $\int_{-\infty}^{\infty} \rho(x)dx = 1$.

3. $\text{Prob}(a \leq x \leq b) = \int_{a}^{b} \rho(x)dx$.

The distribution function associated with the density $\rho(x)$ is defined by

$$F(x) = \int_{-\infty}^{x} \rho(t)dt.$$ 

It has the properties:

1. $F(s) = \text{Prob}(x \leq s)$.

2. $\lim_{x \to -\infty} F(x) = 0$

3. $\lim_{x \to \infty} F(x) = 1$

4. $F(x)$ is an increasing function of $x$.

5. $\frac{dF}{dx} = \rho(x)$

The mean and variance of a density $\rho(x)$ are defined by

$$\mu = \text{Mean}(x) = \int_{-\infty}^{\infty} x\rho(x)dx$$

and

$$\sigma^2 = \text{Var}(x) = \int_{-\infty}^{\infty} (x - \mu)^2 \rho(x)dx$$

The standard deviation, $\sigma$, is the square root of the variance.
1.2 Uniform Density

The \textbf{uniform density} on the interval \([a, b]\) is defined by

\[
\rho(x) = \begin{cases} 
0 & x < a \\
\frac{1}{b-a} & a \leq x \leq b \\
0 & x > b
\end{cases}
\]

An important special case is the uniform density on \([0, 1]\)

\[
\rho(x) = \begin{cases} 
0 & x < 0 \\
1 & 0 \leq x \leq 1 \\
0 & x > 1
\end{cases}
\]

This density has mean \(\mu = 1/2\) and variance \(\sigma^2 = 1/12\).

1.3 Normal Density

The \textbf{normal density} with mean \(\mu\) and standard deviation \(\sigma\) is defined by

\[
\rho(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

The case \(\mu = 0, \sigma = 1\) is especially important

\[
\rho(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}
\]

2 Random Number Generators

Computer algorithms for generating random numbers are \textit{deterministic} algorithms. Although the sequence of numbers produced by a random number generator appears random, the sequence of numbers is completely predictable and for this reason they are often called \textbf{pseudo-random}.

Since computers have only a finite number of different states, the sequence of pseudo-random numbers produced by any deterministic algorithm must necessarily repeat itself. The number of random numbers generated before the sequence repeats is called the \textit{period} of the generator.

A good random number should have the following characteristics:
1. **Randomness.** It should pass statistical tests of randomness.

2. **Long Period.** For obvious reasons.

3. **Efficiency.** This is important since simulations often require millions of random numbers.

4. **Repeatability.** It should produce the same sequence of numbers if started in the same state. This allows the repetition and checking of simulations.

Almost all random number generators used in practice produce uniform \([0,1]\) distributed random numbers and from these random numbers with other distributions can be produced if required.

### 2.1 Algorithms

There are many algorithms for generating pseudo-random numbers.

#### 2.1.1 Linear Congruential Generators

The simplest are the **linear congruential** generators. Starting with the seed \(x_0\), these generate a sequence of integers by

\[
x_{k+1} = (ax_k + b) \mod M
\]

where \(a\), \(b\) and \(M\) are given integers. All the \(x_k\) are integers between 0 and \(M - 1\). In order to produce floating point numbers these are divided by \(M\) to give a floating point number between 0 and 1. The quality of a linear congruential generator depends on the choice of \(a\), \(b\) and \(M\), but in any case the period of such a generator is at most \(M\) (why?).

Here is a Scilab function to produce a sequence of \(n\) pseudo-random numbers from a congruential generator starting with seed \(x0\):

```scilab
def function x = crand(a, b, m, x0, n)
    x = zeros(1, n)
    x(1) = x0
    for i = 2:n
        x(i) = pmodulo(a*x(i-1)+b, m)
    end
    x = x/m
endfunction
```
Here is a simple comparison with Scilab’s random number generator:

\[
\text{--> } \text{xx} = \text{c} \text{rand}(125, 3, 8192, 1234, 1000);
\]

\[
\text{--> } \text{yy} = \text{rand}(1, 1000);
\]

\[
\text{--> } \text{subplot}(2, 1, 1)
\]

\[
\text{--> } \text{histplot}(100, \text{xx})
\]

\[
\text{--> } \text{subplot}(2, 1, 2)
\]

\[
\text{--> } \text{histplot}(100, \text{yy})
\]
2.1.2 Fibonacci Generators

Another common type of generator are the Fibonacci generators. A typical example generates a sequence by

\[ x_k = (x_{k-17} - x_{k-5}) \mod 1 \]

This example uses the previous 17 numbers (and needs 17 seeds to get started). It has a number of advantages compared to congruential generators:

1. The \( x_k \) are floating point numbers, so no division is needed.
2. It has a much longer period since the repetition of a single number does not entail repetition of the whole sequence.
3. It can have much better statistical properties.

2.1.3 Practical Generators

The random number generators used in practice need to have very well tested statistical properties, have long period, and be very efficient. A common choice is a combination of a number of congruential and/or Fibonacci generators.

3 Scilab

Scilab has two random number generators \texttt{rand} and \texttt{grand}. We will only discuss the simpler generator \texttt{rand} which can produce either uniform \([0, 1]\) or normal \( \mu = 0, \sigma = 1 \) random numbers. Uniform random numbers are the default.

Typical uses of \texttt{rand} are:

1. \texttt{rand(m, n)} – gives a \( m \times n \) matrix of random numbers.
2. \texttt{rand(a)} – for matrix \( a \) gives a matrix of random numbers the same size as \( a \).
3. \texttt{rand(m, n, 'normal')} – gives a \( m \times n \) matrix of normally distributed random numbers.
4. `rand('seed', 0)` – resets the random number generator to its original state.

Here are some examples:

```matlab
--> a = rand(4, 4)
   a =
   ! 0.2113249  0.6653811  0.8782165  0.7263507 !
   ! 0.7560439  0.6283918  0.0683740  0.1985144 !
   ! 0.0002211  0.8497452  0.5608486  0.5442573 !
   ! 0.3303271  0.6857310  0.6623569  0.2320748 !

--> b = rand(a)
b =
   ! 0.2312237  0.3076091  0.3616361  0.3321719 !
   ! 0.2164633  0.9329616  0.2922267  0.5935095 !
   ! 0.8833888  0.2146008  0.5664249  0.5015342 !
   ! 0.6525135  0.312642  0.4826472  0.4368588 !

--> c = rand(a, 'normal')
c =
   ! -1.3772844  -0.3888655  1.1323911  0.3268713 !
   ! -0.1728369  0.6543045  -0.2330131  0.5613533 !
   ! -0.6019869  -0.7004486  -0.2343923  -0.1890526 !
   ! -1.5619521  -0.8262233  1.4027611  0.4249849 !

--> rand('seed', 0)

--> d = rand(c)
d =
   ! 0.2113249  0.6653811  0.8782165  0.7263507 !
   ! 0.7560439  0.6283918  0.0683740  0.1985144 !
   ! 0.0002211  0.8497452  0.5608486  0.5442573 !
   ! 0.3303271  0.6857310  0.6623569  0.2320748 !
```

-->d-a
ans =
! 0. 0. 0. 0. !
! 0. 0. 0. 0. !
! 0. 0. 0. 0. !
! 0. 0. 0. 0. !

-->e = rand(1, 10000, 'normal');

-->histplot(100, e)