Lecture 1: Generating random numbers

Fugo Takasu
Dept. Information and Computer Sciences
Nara Women's University
takasu@ics.nara-wu.ac.jp

28 April 2008

1 Random variable

A random variable X has an associated probability with it. X can be either discrete or continuous.

An example of a discrete random variable is toss of coin (0 for tail and 1 for head with equal probability 1/2), dice (1 through 6 with equal probability 1/6), or anything that follows binomial, geometric and poisson distribution. Let us now denote the probability of drawing a certain value, say i, as P_i. Then it satisfies the following properties.

$$0 \leq P_i \leq 1$$
$$\sum_i P_i = 1$$

where summation is taken for all possible i. P_i is called probability distribution.

An example of a continuous random variable is the time for which you must be patient in a waiting line, or anything that follows uniform, exponential and gaussian (normal) distribution. For continuous random variable X, the probability that X lies within a short interval x and $x + dx$ is defined as

$$\text{Prob}[x < X < x + dx] = P(x)dx$$

where $P(x)$ is called probability density function (pdf). It satisfies

$$0 \leq P(x)$$
$$\int P(x)dx = 1$$

where integral is taken for possible range of x.

Once a probability distribution (for discrete) or a probability density function (for continuous) is given, the expected value (mean, average) and the variance of the random variable are calculated as follows.

For discrete random variable,

$$\mathrm{E}[X] = \sum_i i \times P_i$$
$$\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \sum_i (i - \mathrm{E}[X])^2 \times P_i$$

For continuous random variable,

$$\mathrm{E}[X] = \int x \times P(x) dx$$
$$\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \int (x - \mathrm{E}[X])^2 \times P(x) dx$$

Variance can also be calculated using the identity

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$$

Here below I list some continuous random variables often used in many sciences.

1.1 Uniform random variable

An uniform random variable X takes a value x within a certain range with equal probability. Let the range be $a < x < b$. Then the probability density function (pdf) is given as

$$P(x) = \frac{1}{b-a} \quad \text{for} \quad a < x < b$$
$$= 0 \quad \text{otherwise}$$

And naturally $\int_a^b P(x)dx = 1$. Derive the expected value and the variance. Try.

1.2 Exponential random variable

An exponential random variable X takes a non-negative value x ($0 < x < \infty$). The pdf contains a parameter $\gamma > 0$ and it is defined as

$$P(x) = \gamma \exp[-\gamma x]$$

Confirm that $\int_0^\infty P(x)dx = 1$. Derive the expected value and the variance.

1.3 Gauss random variable

A gauss random variable (Gaussian) X takes a value x ($-\infty < x < \infty$). The pdf is

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-m)^2}{2\sigma^2}\right]$$

and it contains two parameters m and σ. This pdf is often presented as $N(m, \sigma^2)$. m is the expected value and σ^2 is the variance.

1.4 Exercise

1. Derive the expected value and the variance of an uniform random variable X whose range is $a < X < b$.
2. Derive the expected value and the variance of an exponential random variable X whose pdf is given as $P(x) = \gamma \exp[-\gamma x]$.
3. Check that the expected value and the variance of a random variable X that follows $N(m, \sigma^2)$ is m and σ^2, respectively, i.e., $\int_{-\infty}^{\infty} xP(x)dx = m$ and $\int_{-\infty}^{\infty} x^2P(x) - m^2 = \sigma^2$.

2 Generating random numbers

Uniform random variable is very important because many other random variables can be derived and transformed from uniform random variable. Let us now focus on an uniform random variable whose range is $0 < x < 1$.

2.1 Generating uniform random numbers

There are several algorithms to generate uniform random numbers. But some of them produce rather poor quality (e.g., built-in random number generator in C like `rand()`). In this lecture we will use an algorithm called **Mersenne twister** which has been widely used a recommended random number generator.

Note that any random number generated by some algorithmic method is nothing more than pseudo random number. In this lecture we will not step into a philosophically deep question "Can any algorithm produce real random number?".

To generate random numbers in most algorithms including Mersenne twister, we need a "seed" to specify a series of random numbers. If we start generating a series of random numbers with the same seed, we have always the same series of random numbers.

Example of series of random numbers generated using the same seed value.

Trial 1: $\{0.201669, 0.411992, 0.146775, 0.236965, 0.773501, \dots\}$

Trial 2: $\{0.201669, 0.411992, 0.146775, 0.236965, 0.773501, \dots\}$

Trial 3: $\{0.201669, 0.411992, 0.146775, 0.236965, 0.773501, \dots\}$

This doesn't make sense in simulation except for debugging program. For choosing a different seed value every time simulation is run, the time of run (day, hour, minute, second, which can be readily obtained in C program) can be a good seed to produce different series of random numbers.

Here is an example how to generate uniform random number ($0 < x < 1$) in C.

```
#include <stdio.h>
#include <time.h>  /* This is necessary to get time */

/* functions of Mersenne twister */
extern void init_genrand(long);
extern double genrand_real1(void), genrand_res53(void);

int main (int argc, const char * argv[]) {

  unsigned long seed;  /* the seed is unsigned long */
  int i;
  double rand;
  FILE *fp;

  seed = (long)time(NULL);
  printf("Hello, World! The seed is %ld\n", seed);
  fp = fopen("uniform.out", "w");

  init_genrand(seed);  /* initialize the random number generator with "seed" */

  for(i=0; i<100; i++){
    rand = genrand_real1();
    //rand = genrand_res53();
    printf("%.20f\n", rand);
    fprintf(fp, "%20f\n", rand);
  }

  fclose(fp);

  return 0;
}
```

2.2 Checking the uniform random numbers

Once we obtain a series of numbers generated by a generator, we would be tempted to see the quality of randomness. To test if the obtained series is uniform random number of good quality (statistically, this is equivalent to test the null hypothesis "The series constitutes uniform random number" using Kolmogorov-Smirnov test). Refer to text of statistics.

In this lecture we go much easy as to the quality of random number generated, i.e, just a visual inspection. We use *Mathematica* to visualize the obtained series. We 1) read a set of random numbers saved in a file, 2) count the number of random numbers within a certain interval, and 3) make the frequency distribution to see if generated random numbers reflect original pdf.

In the notebook, we first read a package to do statistics, Statistics`DataManipulation` at the beginning. This should be carried out only once after *Mathematica* is launched. We then designate the directory where our target file resides using the command SetDirectory. Note that *Mathematica*'s commands start with capital letter and they are case sensitive. Below is a sample *Mathematica* notebook.

2.3 Exercise

1. Complete a C program that generates 100 uniform random numbers. Write the series of random numbers into a file.
2. By viewing the saved file, check and judge if these numbers follow uniform distribution. Does the simulated distribution look uniform if we increase the number of random numbers to 1000, or more?

3 Generating exponential random number

If an uniform random variable $0 < x < 1$ is given, exponential random variable is readily derived. We define a new random variable $y = -\frac{\ln x}{\lambda}$. Then y is exponentially distributed with $\gamma = \lambda$. This can be shown as follow. Let $p(x)$ be the pdf of a uniform random variable x and $q(y)$ be the pdf of y.

$$\begin{aligned} p(x) &= 1 \quad \text{for} \quad 0 < x < 1 \\ &= 0 \quad \text{for otherwise} \end{aligned}$$

From the basic rule of variable transform, $|q(y)dy| = |p(x)dx|$, we see

$$\begin{aligned} q(y) &= p(x)\left|\frac{dx}{dy}\right| \\ &= p(x)\lambda \exp[-\lambda y] \\ &= \lambda \exp[-\lambda y] \end{aligned}$$

thus, exponential random variable is easily obtained from an uniform random variable. This transformation has been implemented as expdev() which returns an exponential random number with $\gamma = 1$.

4 Generating gaussian random numbers

With the transformation of two uniform random variables x_1, x_2, $(0 < x_1, x_2 < 1)$,

$$\begin{aligned} y_1 &= \sqrt{-2\ln x_1}\cos 2\pi x_2 \\ y_2 &= \sqrt{-2\ln x_1}\sin 2\pi x_2 \end{aligned}$$

y_1 and y_2 both are gaussian random variable with $m = 0$ and $\sigma = 1$. This transformation has been implemented as gasdev() and is available for use.

Gaussian with mean m and variance σ^2 is often written as $N(m, \sigma^2)$. Let x be a random variable of $N(0, 1)$. Then $y = a + bx$ is random variable of $N(a, b^2)$. That is, any gaussian random variable is derived from $N(0, 1)$ with simple transformation.

5 Random numbers generated by C

We have learnt how to generate random numbers using Mersenne twister. Random number generator genrand_real1() returns a double number uniformly distributed between 0.0 and 1.0. The genrand_real1() is

used to generate other random number expdev() that returns an exponential distributed number with $\gamma = 1$, and gasdev() a gaussian with $m = 0$ and $\sigma = 1$. Here is a sample program to generate these three random numbers and write them to files.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define STEP 100

extern void init_genrand(long);
extern double genrand_real1(void);
extern double expdev(void), gasdev(void);

int main(void)
{
  long seed; /* seed is long integer */

  double rand1, rand2, ran3;
  int i;
  FILE *fp_uniform, *fp_exp, *fp_gauss;

  seed = (long)time(NULL);   /* seed should be chosen different at every new  trial */

  /* Initialize the random generator ran2 with NEGATIVE argument */
 init_genrand(seed);

  printf("Seed is %ld\n", seed);
  fp_uniform = fopen("uniform.out","w");
  fp_exp = fopen("exp.out", "w");
  fp_gauss = fopen("gauss.out", "w");

  for(i=0; i<STEP; i++){

    rand1 = genrand_real1();
    rand2 = expdev();
    rand3 = gasdev();

    printf("%f %f %f\n", rand1, rand2, rand3)
    fprintf(fp_uniform, "%f\n", rand1);
    fprintf(fp_exp, "%f\n", rand2);
    fprintf(fp_gauss, "%f\n", rand3);

  }

  fclose(fp_uniform);
  fclose(fp_exp);
  fclose(fp_gauss);

  return 0;
}
```

6 Checking generated random numbers

The above sample program generates 100 random numbers (uniform, exponential, and gaussian) and writes these into three separate files. We now want to check if these generated numbers reflect uniform, exponential, and gauusian random numbers. We use *Mathematica* to do this.

6.1 Checking the exponential random numbers

We do the same to see if generated exponential random numbers are really distributed "exponentially".

6.2 Exercise

The function expdev() returns an exponential random number of $\gamma = 1$. As already shown simple transformation expdev()/γ gives a random number whose pdf is $\gamma \exp[-\gamma x]$. Using this relationship, generate exponential random numbers of $\gamma = 2$ and confirm that the frequency distribution of these numbers coincides with $\gamma \exp[-\gamma x]$ where $\gamma = 2$.

6.3 Checking the gaussian random numbers

Same above.

6.4 Exercise

The function gasdev() returns a gaussian random number of $m = 0$ and $\sigma = 1$. As already shown, simple transformation $y = ax + b$ where x is $N(0, 1)$ and a and b are constant gives a random number of $N(b, a^2)$. Using this relationship, generate gaussian random numbers of, say, $m = 1$ and $\sigma = 2$ and confirm that the frequency distribution of these numbers coincides with $N(1, 4)$.

7 Summary

We are now a bit confident that we are good at generating random numbers in C programming and that these random numbers are of good quality (we hope so). We will use these to simulate a variety of stochastic processes in the course.