Discrete Mathematics

Computer Science 2430

Programming project 4

For this assignment, write, test, and execute code to solve the following problems. You should also answer all of the questions.

Here we will be examining the ways that various random numbers interact in systems. This is a significant problem in big data analysis. Often times, when we are analyzing large data sets, we are trying to correlate outputs with one or more inputs. In real life, the outputs that we are analyzing depend on a large number of inputs. However, one common initial assumption in big data analysis is to treat the data as if it were dependent on the one variable we are analyzing along with a random signal superimposed on that from all of the other inputs. For this project, we will be looking specifically at two types of random noise, linear random and normally distributed (or Gaussian) random noise. The linear random function takes random variables evenly distributed between two points while the Gaussian distribution is the classic bell curve (https://en.wikipedia.org/wiki/Normal\_distribution) (don’t worry about the mathematics on this page, this is just to illustrate the graph).

In terms of big data analysis, a flat distribution is typical when I’m analyzing one input and the input(s) I’m ignoring don’t have any bearing on my result. On the other hand, a normal, or Gaussian distribution is one type of result I can see when the other input’s I’m ignoring do have an impact on the result.

Part 1

Use a linear random number generator (such as Java’s Math.random()) function to simulate random noise. Write a function that takes a single integer from 0 to 100 as an argument and returns true that percent of the time. Write a driver which calls your function a large number of times and keeps track of what percentage of the time it returns true.

1. If I were to call your random function once with an argument of 70 it would obviously either return true or false, i.e. it would either be true 100% of the time or 0% of the time. If I called it twice it would either return true twice (true 100% of the time), it would return true once and false once (true 50% of the time), or it would return false twice (true 0% of the time). From your data, about how many times would I need to call your function to reliably have the overall percentage returned be about equal to 70% (say somewhere between 69% and 71%)?
   1. Running it 100,000 times, you would have to call a Linear Distribution 132.6 times on average.
2. How long would I have to run your driver for the overall percentage returned to be between 69.9% and 70.1%?
   1. Running this program 100000 times, you would need to, on average,run it 1103.0 times for Linear Distribution
3. What does this say about the overall accuracy of data with a random linear signal superimposed on it? This result is correlated (in simple models) with the number of samples I have to have in my data set in order to derive meaningful results.
   1. It is fairly consistent, for a 10X increase in precision, it would require a 10X increase in number of calls

Part 2

Now we will do much the same with Gaussian random distributions. For this portion you will need to use a Gaurssian random number generator (such as Java’s Random.nextGaussian()).

While a linear random number is evenly distributed over some range, the Gaussian distribution is described by two numbers, the mean (the average value) and the standard deviation (a measure of how spread out the bell curve is). Small standard deviations give very narrow bell curves, broad standard deviations give wide bell curves.

Write a function and/or modify a copy of your driver to generate random Gaussian numbers with a specified mean and standard distribution. As an example, if I wanted to generate random numbers with a mean of 70 and a standard deviation of .5, I could code

double gausRand = myRand.nextGaussian() \* 0.5 + 70. For this round of tests, you should treat a test as true if it returns a value within the specified range and false otherwise. For example, if my function were calculating Gaussian values with a mean of 70 and a standard deviation of .5, it would return true if the number generated were 70.43 but would return false if the number generated were 70.51.

1. Running it 100,000 times, you would have to call a Gaussian Distribution 925.0 times on average.
2. Running this program 100000 times, you would need to, on average,run it 2551.4 times for Gaussian Distribution
3. It is not as consistent, 10X increase in precision resulted in an almost 20X increase in calls

7, 8, and 9) Answer the questions from part 1 using your Gaussian random numbers with with a mean of 70 and a standard deviation of 2.

1. Running this program 100000 times, you would need to, on average,run it 921.95048 times for Gaussian Distribution.
2. Running this program 100000 times, you would need to, on average,run it 2620.14727 times for Gaussian Distribution.
3. This has almost the same results as the smaller deviation, a 10X increase in precision results in a 20X increase in calls
4. Essentially, questions 1, 2, 4, 5, 7, and 8 had you compare the number of times that you needed to run your code to go from an answer accurate to +/- 1% to an answer accurate to +/- .1%. If you did not know the type of noise (or random data) superimposed on a good signal, what rules could you give for helping to determine if your answer is accurate?
   1. How consistent it is when going from one precision to the other, if it always scales consistently with the number of times you are comparing it.

Part 3

Now we will use your driver from part 1 to derive some rules about how linear random distributions interact. Use your driver to calculate how often the following two statements return true

(myRandomFunction(70) && myRandomFunction(40))

(myRandomFunction(70) || myRandomFunction(40))

. The first function should be true about 28% of the time (.7\*.4), while the second function should be true about 82% of the time (1-(1-.7)\*(1-.4)).

11) How many times did you need to run each test to get to an accuracy of about +/- 1%? 12) How many times did you need to get an accuracy of about +/-.1%?

13) How do these numbers compare to your answers for questions 1 and 2? Why do you think this might be the case?