Software Development (cs2500)

Lecture 9: A Trip to the Casino

M.R.C. van Dongen

October 15, 2010

Contents

1 Overview

These lecture notes do not correspond to the book.

2 Monte Carlo Integration

2.1 Basic Idea

This section studies another application of random numbers: probabilistic integration. The method which we shall study is called *Monte Carlo integration* and we shall study it to approximate the area of the unit circle. Before doing this, we shall study the general idea.

Throughout these notes we shall make the same assumptions about "randomness" as before, i.e. if numbers are random then the probability of a given number to appear at a certain position in the random sequence is equal for each number.

In the following we have two shapes: an inner shape I and an outer shape O. The area $A(O)$ of O is known, but we do not know the area $A(I)$ of I. However, we do have a method to generate random coordinates (dart positions) inside O and we know when a dart is inside I. The probability that a random dart position is inside I should be equal to $A(I)/A(O)$. (We can get such probabilities if O is a rectangle and the x and y coordinates are selected uniformly from the range of allowed values. For example, if O is a square and I is the lower third of O, then the probability that a random dart is inside I should be equal to $1/3$.

We now start throwing n darts at random positions in O. When we're done, we count the number of darts, i, that are inside I. If n is sufficiently large, then it can be shown that the ratio i/n is a good approximation of the ratio $A(I)/A(O)$, i.e. $i/n \approx A(I)/A(O)$. Since we know $A(O)$, n, and i, this gives us

$$A(I) \approx A(O) \times i/n\,. \tag{1}$$

2.2 The Area of a Circle

We shall now apply the ideas from the previous section by approximating the area of the *unit circle*. Here, the *unit circle* is the circle with centre $(0,0)$ and radius 1. For convenience we let the "*extended unit square*" be our outer shape O: it is the square with lower left coordinate $(-1,-1)$ and upper right coordinate $(1,1)$. It follows that $A(O) = 4$. Figure 1 depicts I and O.

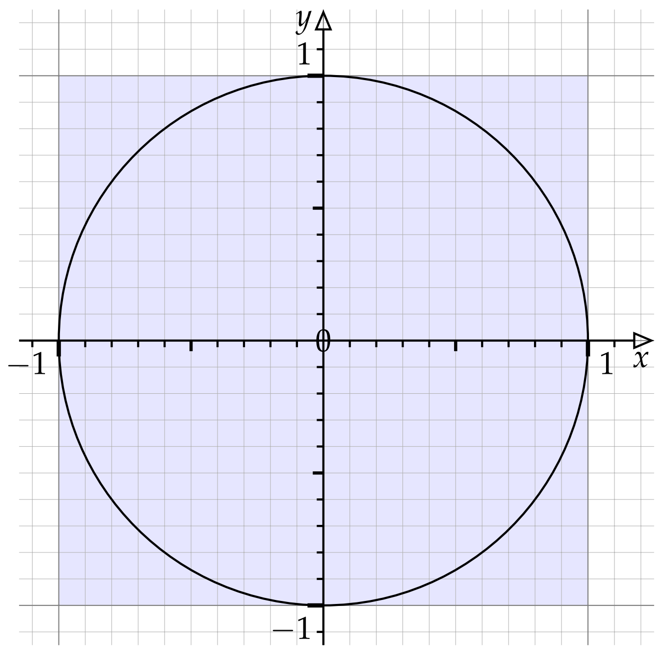

Figure 1: Unit circle, I, and the extended unit square, O. The area inside O is filled in blue.

We now throw $n = 1000$ "darts" at random positions inside O. Figure 2 depicts the situation after the throwing of the darts. After careful counting we find that $i = 793$ darts are inside the unit circle. Plugging $A(O) = 4$, $n = 1000$, and $i = 793$ into Equation 1 gives us $A(I) \approx 4 \times 793/1000 = 3.172$, which is quite good since it is well known that $A(I) = \pi$.(This follows from the fact that a circle with radius r has area πr^2.) Choosing larger and larger values for n and repeating the experiment should lead to more and more accurate approximations.

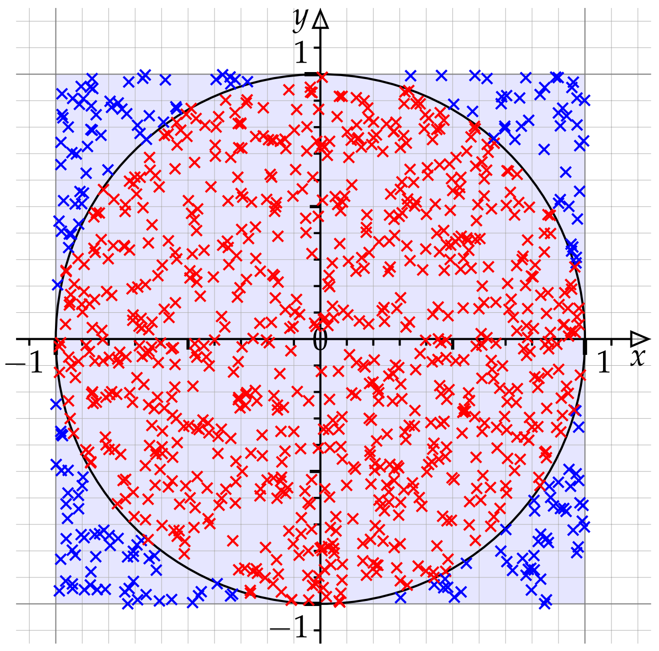

Figure 2: Situation after throwing 1000 darts at random positions inside the extended unit square. A cross indicates the position of a dart. Darts that are inside the circle are red. The remaining darts are coloured blue.

2.3 Implementation

A possible implementation is presented in Figure 3. The modifier '`final`' in the declarations tells the `Java` compiler that the assignments to the variables are final. Any subsequent assignment to these variables will now result in an error.

3 For Wednesday

Study the notes and read Chapter 3.

```
import java.util.Random;
public class MonteCarlo {
    public static void main( String[] args ) {
        final Random rand = new Random( );
        final int totalSamples = 1000000; // Total number of random points.
        final double totalArea = 4.0;     // Total area of extended square.

        int hits = 0; // Initialise number of random points inside circle.
        for ( int sample = totalSamples sample - != 0; ) {
            // Generate random point.
            double x = 2.0 * (rand.nextDouble( ) - 0.5);
            double y = 2.0 * (rand.nextDouble( ) - 0.5);
            // Increment hits if point is inside circle.
            hits += x*x + y*y <= 1.0 ? 1 : 0;
        }
        // Compute approximation of area of circle.
        final double ratio = (double)hits / totalSamples;
        final double approximation = totalArea * ratio;
        System.out.println( ”PI = ” + Math.PI );
        System.out.println( ”PI ~ ” + approximation );
    }
}
```

Figure 3: Approximating π with Monte Carlo integration.