Chapter 7

The Monte Carlo Method

The "Monte Carlo method" is a general probabilistic algorithm for estimating the probability of an event based on the a number of random trials.

7.1 Randomized Algorithms and Probability

A *randomized algorithm* is an algorithm that makes at least one decision based upon a random choice, rather than a fixed rule.

At first glance, it may seem surprising (or even unbelievable) that incorporating an element of chance into an algorithm can result in a correct and efficient algorithm, but as we will see in this chapter, there are many situations where randomness is extremely effective.

Randomized algorithms generally depend upon the laws of probability, and therefore before we begin studying randomized algorithms, it makes sense to formally define some of the terms and concepts of probability.

- An **experiment** is any procedure that can be repeated any number of times, and has a well-defined set of outcomes. (In order to be interesting from a statistical perspective, the result of the procedure is generally not known in advance.)

- An **sample outcome** (often denoted s) is a member of the set of outcomes of an experiment.

- The **sample space** (often denoted S) is the complete set of all possible sample outcomes of an experiment.

- An **event** is any set of sample outcomes. When the outcome of an experiment is a member of this set, the event is said to **occur**.

- Two events are **independent** if the occurance of one of the events is not related to the occurance of the other. For example, if we flip a fair coin twice, then the event "first flip is heads" is independant of the event "second flip is heads". However, the events "first flip is heads" and "first flip is tails" are *not* independent, because the occurance of one of these events depends on whether the other has occured. In this case, the correspondance is absolute– but this is not always the case.

- A **trial** is the actual "execution" of an experiment. For example, the procedure of tossing a coin and observing whether the result is heads or tails is an experiment, but the act of actually flipping the coin is a trial.

- A **population** is the set of all unique trials. This set may be infinite. For example, the population of all coin tosses is infinite, because we can perform as many unique trials as we like by flipping the coin repeatedly. In other situations, the population may be finite– for example, if our experiment is to ask a person enrolled in S-Q whether their last name contains the letter 'e', then the set of unique trials is the set consisting of one trial for each person enrolled in the course.

- A **sample** is a set of trials. A **random sample** is a set of trials selected at random from the population.

For example, consider the act of rolling a 6-sided die. The procedure consisting of rolling the die and reporting which face is visible is the experiment; and each time we actually perform this procedure we are performing a trial. We can roll the die as often as we like, and every time we roll the die the outcome (i.e. which side of the die is showing) must be one of the 6 sides.[1] Each possible side of the die is a possible sample outcome, and these six possibilities constitute the entire sample space.

We can define a number of possible events for this experiment– for example, rolling any particular number, or rolling an odd number, or rolling a number less than four.

[1]We assume that the die must always come to rest on one of its faces, not balanced on an edge or a corner.

7.2 Monte Carlo Simulation

Monte Carlo simulations are one of the most common randomized algorithms. A Monte Carlo simulation is a method of estimating the value of an unknown quantity by making use of the principles of inferential statistics.

In brief (since this is a course in algorithms, not probability and statistics), the guiding principle of inferential statistics is that a random sample tends to exhibit the same properties as the population from which it is drawn.

For example, if you flip a coin 10 times, and it comes up heads every single time, then you might start to form an opinion that the coin is not fair, and that it is biased towards coming up heads– you might begin to believe that the population of all possible flips of this coin contained more heads than tails.[2] If you flip the same coin 100 more times, and it still comes up heads every single time, it would probably be hard to convince you that the coin was fair at all (in fact, you might start to believe that it had two heads!).

On the other hand, if you flipped the coin 100 times and roughly half of the time it came up heads, then you would probably believe that it was a fair coin– or at least not unfair to any obvious degree.

In both cases, your belief whether the coin is fair or not is based on the intuition that the behavior of a sample of 100 flips is similar to the behavior of the population of all flips of your coin. Luckily, this belief is not without basis in fact– and we can make use of this fact to achieve good estimates of probabilities with relatively little effort. This method is formalized in algorithm 7.1, which shows a method of estimating p for a coin, based on a set of random trials.

7.2.1 Accuracy of a Monte Carlo Simulation

When we use algorithm 7.1 to perform a Monte Carlo simulation, we are trusting that fate will not give us an unusual or freakish sample– we are trusting that the true probability is reflected in our observations. In the long run, we can generally rely upon this assumption– although it is important to know how many samples we will have to take before this assumption holds.

While we trust in luck to ensure that our sample is not wildly unusual, we must also realize that it is extremely unlikely that fate will provide a sample that is *exactly* like the

[2] From a mathematical perspective, it is incorrect to so blithely make a comparison of the cardinality of these two infinite sets. Although most people's intuition about the nature of infinity is far from correct, most people do find probability reasonably intuitive, and I appeal to that intuition here instead of embarking on a lengthy mathematical digression.

Algorithm 7.1 (Monte Carlo Simulation) *An algorithm to estimate the probability p that a coin will come up heads.*

- Flip the coin N times. Count the number of times heads appears, and call this number M.
- As N grows larger and larger, we expect that, through the graces of probability, the ratio M/N will become very close to the actual ratio p.

  Thus we estimate p as (M/N).

population.[3] Consider the example of flipping a fair coin 1000 times. While it is likely that the number of heads and tails will be close to 500, it is also unlikely that they will both be *exactly* 500. What is far more likely is that the number of heads will fall in some range around 500.

This leaves us with two questions to ponder:

1. How many samples do we need to take in order to safely assume that we have taken "enough" samples?
2. How accurate do we believe that our estimate is, given a certain number of samples?

As you might guess, the answers to these two questions are related– the more sure we want to be that our sample is good, the more samples we need to take, and the more samples we take, the more likely it is that we will get a more accurate estimate.

Unfortunately, it is never possible to achieve perfect accuracy through sampling unless you sample the entire population– no matter how many samples you take, we can never be sure that the sample set is typical until we verify every last element of the population (and since we are usually dealing with infinite populations, this is simply impossible). Of course, this is not to say that our estimates are not perfectly correct– we might arrive at the exactly correct value, but the trouble is that *we can't know that our estimate is correct*! All we can measure is how *likely* it is to be correct.

[3] In some cases, such as we will explore later in the chapter, this is not even *possible*, no matter how kindly fate smiles upon our program– our estimates will always be rational numbers, and not all probabilities are rational.

Since this is a course in data structures and algorithms, and not a course in statistics, rather than deriving the relationship between the sample size and accuracy estimate from first principles (which requires the first week or two of a course in mathematical statistics to accomplish), it is now time to begin making gross (but useful) generalizations.

The relationship between the sample size and our estimate of the accuracy of an estimate arrived at from a sample is given by the following equations:

$$e = \frac{z}{2\sqrt{n}} \tag{7.1}$$

$$n = \left(\frac{z}{2e}\right)^2 \tag{7.2}$$

In these equations, n is the sample size, and e is the "margin of error", which is the fraction by which we believe our estimate may be incorrect. The variable z is a function of how *confident* we desire to be in our estimate of e– the probability that our estimate is within e of the true value. Computing z is a laborious process (and it was even more laborious before computers were given this task). However, the following table contains values of z for some common confidence levels:

z	Confidence Level
1.96	95%
2.58	99%
3.29	99.9%

A few very important caveats about these equations:

- They should not be used with small n. The assumptions upon which they are based break down when n is less than 30.
- Similarly, they should not be used with probabilities that are extremely near zero or one unless a large number of samples are drawn. One rule of thumb is that the estimate should be based on at least 5 trials with both outcomes– so if you are estimating the probability of an event that has a very low true probability, you may have to take a large number of samples before you have any evidence at all that the probability is non-zero– but if you happen to draw a positive sample in one of the first trials and stop soon thereafter, your probability estimate may be wildly high.

- These equations are pessimistic. Assumming that the previous two conditions are met, they generally give margins of errors that are too wide (or suggest that you should perform more trials than you really need to). Personally, this is the direction I prefer to err in– I would rather believe that my estimate is less accurate than it is, instead of thinking that it is more accurate than the facts would support. However, if you are trying to perform the absolute minimum number of trials necessary to achieve a given level of confidence, you may wish to find a tighter bound.

Computing the Margin of Error

Returning to our example of flipping a coin 1000 times, if the coin comes up heads exactly 500 times, our estimate of the probability of the coin coming up heads is 0.5. However, now we can also compute our "margin of error" for this estimate. With 95% confidence, we can say that our estimate is accurate to within 3.1%:

$$e \;=\; \frac{z}{2\sqrt{n}} \;=\; \frac{1.96}{2\sqrt{1000}} \;=\; 0.031$$

Similarly, we can compute our "margin of error" with a 99% level of confidence, using the same method, but with a z of 2.58:

$$e \;=\; \frac{z}{2\sqrt{n}} \;=\; \frac{2.58}{2\sqrt{1000}} \;=\; 0.041$$

Notice that our margin of error has increased. In order to gain more confidence in our estimate, we have had to widen the margin. Note that the *only* way to achieve 100% confidence is to widen the margin of error to be so wide that it spans the entire range from 0.0 to 1.0– the only way that we can be 100% confident that our margin of error contains the correct probability is to include all probabilities!

Returning to the coin example, if our estimated p is 0.5, and our margin of error (based on 1000 trials, at a 95% level of confidence) is 0.031, then we can restate our estimate as 0.5 ± 0.031 (with a 95% level of confidence). This way of expressing the estimate together with the margin of error is called a *confidence interval*.

In this case, based on our 1000 trials, we can say that we are 95% confident that the true probability of the coin coming up heads is within the range from 0.469 to 0.531. The true probability *might* be outside this range, but we are 95% confident that it is not.

Note that if our margin of error is large enough (either because we are requiring a very high confidence level, or the number of samples is too small), the confidence interval may include values outside the range from 0 to 1. These values should be discarded, because they are not valid probabilities.

Computing the Number of Trials

In the last section, we computed the confidence interval for a sample of 1000 trials. In this section, we will ask the problem from the other perspective– in order to get an estimate of the probability p of a coin (i.e. the probability that the coin will come up heads) that is accurate to within 5%, with a 95% level of confidence, how many trials do I need to perform?

Returning the equation:

$$n = \left(\frac{z}{2e}\right)^2 = \left(\frac{1.96}{2(0.05)}\right)^2 = \left(\frac{1.96}{0.1)}\right)^2 = (19.6)^2 = 384$$

Therefore, we can be 95% confident that an estimate of p based on 384 coin flips will be accurate to within 5% of the true p. (If we want to be more confident, or to achieve a more accurate estimate with the same level of confidence, then more coin flips will be necessary.)

7.3 Monte Carlo Integration

One good use of the Monte Carlo method is called Monte Carlo integration. The goal of integration is to find the area of a shape X; Monte Carlo achieves an estimate of the area by guessing. The method is almost entirely the same as we saw earlier, except that instead of estimating the p of a coin, we are estimating an area. The trick is to reduce this problem to the problem of estimating p of a coin. Our coin flips are performed by randomly choosing points from an area Y that completely encloses X, and whose area we know– if the point is inside X, then we say that it is "heads", but if it is outside X, then it is "tails". Once we have an estimate on the p for this imaginary coin, we can multiply p by the area of Y to get an estimate of the area of X.

This is described in more detail in algorithm 7.2.

It is important to note that if we cannot find a region Y or a function **inside**(p) that meets the criteria, then this algorithm cannot be used. Metaphorically, this is because we have no way of determining whether the "coin" we just flipped came up heads or tails!

As a concrete example, consider the problem of estimating the area of the quarter circle with a center at zero and a radius of 1. If you recall geometry, you will remember that the area of any circle is πr^2, where r is the radius of the circle, Thus, a complete circle with a radius of 1 will have an area of π, so our quarter circle will have an area of $\pi/4$.

However, if you had forgotten geometry (or simply forgotten what the value of π is), then finding the area of this quarter circle would represent a difficult problem to solve without calculus. Let us imagine, for the moment, that this is the case.

Algorithm 7.2 (Monte Carlo Integration) *An algorithm to find the area of a shape* X.

Let **inside**(x, y) be a function that determines whether point (x, y) is inside shape X or not.

- Find a region Y that completely encloses X, and whose area is known. (For example, Y can be any rectangle that completely surrounds X.)
- Randomly choose N points inside Y, and for each point $t = (x, y)$ use **inside**(t) to determine whether the point p is also inside X, or whether it is inside Y but *not* inside X. Let M be the number of points that fall inside X.
- As N grows larger and larger, we expect that, through the graces of probability, the ratio M/N will become very close to the actual ratio of the area of X to the area of Y.

  Thus we estimate the area of X as (M/N) multiplied by the area of Y (which we already know).

If Y is rectangular, then code to implement this algorithm is shown in figure 7.1.

By consulting a mathematics textbook and looking at the figure, however, you discover the following pieces of useful information:

1. The quarter circle falls entirely in the square with corners (0,0), (0,1), (1,1), and (1,0).

   This defines Y for our purposes.

2. A circle centered at the origin and radius r is defined as the set of points that satisfy the equation $x^2 + y^2 = r^2$. Therefore, since r in our case is 1, then any point such that $x^2 + y^2 \leq 1$ is inside the circle, and all other points are outside it.

   Therefore, for our quarter circle, if $0 \leq x, y \leq 1$, then:

$$\textbf{inside}(x, y) = \begin{cases} \textbf{TRUE if } x^2 + y^2 \leq 1 \\ \\ \textbf{FALSE otherwise}. \end{cases}$$

   This defines the `inside` function for our quarter circle.

This leads directly to the implementation of the `inside` function shown in figure 7.2.

Figure 7.1: Code to perform Monte Carlo integration.

```
double  findArea (int N, double xMin, double xMax,
                double yMin, double yMax)
{
        double x, y;
        double area = (xMax - xMin) * (yMax - yMin);
        int M = 0;
        int i;

        for (i = 0; i < N; i++) {
                x = randomDouble (xMin, xMax);
                y = randomDouble (yMin, yMax);

                if (inside (x, y))
                        M++;
        }

        return ((area * M) / N);
}
```

Figure 7.2: The `inside` function for estimating π, using a circle with radius 1, centered at the origin.

```
int inside (double x, double y)
{

        return ((((x * x) + (y * y)) < 1) ? 1 : 0);
}
```

7.4 Exercises

1. Imagine that you want to estimate the proportion of the population of American voters who consider themselves Democrats. Assuming that you can select randomly from among the voters, and each person you select tells us the truth (two conditions that are hard to satisfy in actual practice), and assuming that both Democrats and non-Democrats are not extremely rare:

   (a) How many voters do you need to ask in order to achieve an estimate that you are 95% confident is within 10% of the true proportion?

   (b) How many voters do you need to ask in order to achieve an estimate that you are 99% confident is within 2% of the true proportion?