Model Markov Processes

In this lesson, we will be discussing the Model Markov Process, which is a completely different kind of distribution. It is neither completely discrete nor continuous.

WE'LL COVER THE FOLLOWING

- Introduction to the Markov Model
  - Some Applications of Markov Model
  - Simplified Version of Gambler's Ruin Example
- Implementation

So far in this course, we've very briefly looked at continuous distributions on doubles and spent a lot of time looking at discrete distributions with small supports.

Introduction to the Markov Model

Let's take a look at a completely different kind of distribution that doesn't easily fit into our "small discrete" bucket but isn't continuous either.

Some Applications of Markov Model

Some motivating scenarios:

- Imagine you find yourself placed at random in a city and you want to take a stroll. At every intersection, you make a random choice of which direction to go next. The random choice of starting points has a particular distribution, and the distribution of possible direction choices could be different at each intersection. The result of each stroll is a path through the city. What is the distribution of possible strolls?
- Imagine a different kind of random walk: you are on a page in a wiki, like Wikipedia or TV Tropes. You read the page and then select at random a link to another page in the site, if there is one. What is the distribution of

possible sequences of pages?

- Imagine you are a gambler who starts with a certain random amount of cash in your pocket and a target for how much you want to win and how much you are willing to lose. If you've won or lost too much, you go home; otherwise, you make a random-sized bet on a slot machine with certain odds, where the distribution of bet sizes and machine choices is a function of how much money you have in your pocket right now. What is the distribution of possible sequences of cash in your pocket?

Each of these cases has several things in common:

1. We start in a randomly chosen state.
2. We randomly move to a new state depending *solely on our current state* and *not* on any "memory" of how we got to that state.
3. The new state might be a "stop state", in which case the sequence of states is over.
4. The question is: what is the distribution of *possible state sequences*?

A random process that has these properties is called a Markov process, after the Russian mathematician **Andrey Markov**. Such processes are beneficial and have been heavily studied for the last hundred years or so.

Andrey Markov

> For this course we are only going to look at Markov processes where each "state change" happens at no particular time; all we care about is the sequence. There is a whole subfield of studying Markov processes

> where we care about how long the gap is between events, and what the distribution of those gaps is; I'm not going to go there.

> **Note:** There is no requirement in a Markov process that any of the distributions involved be discrete. The examples we gave all involved discrete quantities: intersections, web pages, dollar amounts. But you could certainly do a Markov process where you randomly chose a point on a line via some continuous distribution and then chose the next point based on a continuous distribution that is a function of the current position. We will look at examples of such in later lessons; for now, though we'll stick to discrete examples.

> **Note:** There is no requirement whatsoever that the sequence sampled from a Markov process be finite! It can be finite or infinite, depending on the details of the process.

> Suppose we produce a sequence of dice rolls. We roll a die, and no matter what we rolled, we roll it again, every time. Technically that is a Markov process; the next roll depends on nothing, not even the previous roll. We will not be considering such "degenerate" Markov processes in this series; we're interested in processes where the distribution of next states depends on the current state.

Let's temporarily leave aside our discrete weighted distributions and go back to our underlying unweighted `IDistribution<T>` interface for this one. (Working out the "weight" associated with a particular sequence is an interesting problem, but not one that we're going to cover in this series.) Here we have a distribution on sequences of `T`:

```
sealed class Markov<T> : IDistribution<IEnumerable<T>>
{
  private readonly IDistribution<T> initial;
  private readonly Func<T, IDistribution<T>> transition;
```

```
  public static Markov<T> Distribution(
      IDistribution<T> initial,
      Func<T, IDistribution<T>> transition) =>
    new Markov<T>(initial, transition);
  private Markov(
    IDistribution<T> initial,
    Func<T, IDistribution<T>> transition)
  {
    this.initial = initial;
    this.transition = transition;
  }
  public IEnumerable<T> Sample()
  {
    var current = this.initial;
    while(true)
    {
      if (current is Empty<T>)
        break;
      var s = current.Sample();
      yield return s;
      current = this.transition(s);
    }
  }
}
```

> You probably noticed that the state transition function is just another likelihood function: given the current state, it tells you what next states are most likely.

Simplified Version of Gambler's Ruin Example

Let's look at a simplified version of our "gambler's ruin" example.

Our state is the current amount of money we have: an integer. We need a distribution on initial conditions. Let's say we'll start off with exactly n dollars every time, so that's a singleton distribution. If the state ever gets to zero or to twice n, we quit. We represent quitting by producing an empty distribution. We cannot produce any more states because we cannot sample from the empty distribution! If we are not in a quitting state then we flip a coin. This causes our state to transition to either the current bankroll minus one, or plus one.

```
const int n = 5;
var initial = Singleton<int>.Distribution(n);
IDistribution<int> transition(int x) =>
  (x == 0 || x == 2 * n) ?
    Empty<int>.Distribution :
    Flip().Select(b => b ? x - 1 : x + 1);
```

All right, let's take it for a spin:

```
var markov = Markov<int>.Distribution(initial, transition);
Console.WriteLine(markov.Sample().CommaSeparated());
```

The first time we ran this we got:

```
5,4,3,2,3,4,5,4,3,4,3,4,5,4,3,4,5,6,5,
6,5,6,5,6,5,6,5,4,5,6,7,8,9,8,9,8,9,10
```

It took 38 flips to get from 5 to 10. (And we ended up back at the start nine times!)

> **Exercise:** Should the length of this sequence surprise you? Is it longer than average, shorter than average, or does it seem like it should be a pretty typical sample from this distribution? The solution is below.

Show Hint

The easiest way to get a handle on typical sequence lengths is to just draw the histogram:

```
markov.Samples().Select(x => x.Count()).DiscreteHistogram()
```

```
 6|*********************************
 8|************************************
10|****************************************
12|***********************************
14|*****************************
16|**************************
18|************************
20|**********************
22|********************
```

```
24|******************
26|******************
28|******************
30|****************
32|***************
34|***********
36|**********
38|**********
40|********
42|********
44|********
46|******
48|******
50|******
52|****
54|****
56|***
58|****
60|***
62|**
64|**
66|**
68|***
70|**
72|*
```

We cut it off there; the “long tail” in this particular run went out to 208!

We can see from the histogram that the mode is ten flips and most games end in fewer than 38 flips, but there is still a substantial percentage that takes longer.

We seem to be doing pretty well here; we’ve created an entirely new class of distributions that we can explore using fluent programming in only a few lines of code.

Implementation

Let’s have a look at the code:

Program.cs

Bernoulli.cs

BetterRandom.cs

Distribution.cs

Empty.cs

Episode24.cs

Extensions.cs

IDiscreteDistribution.cs

IDistribution.cs

Markov.cs

Projected.cs

Pseudorandom.cs

Singleton.cs

StandardCont.cs

StandardDiscrete.cs

```
using System;
using System.Linq;
namespace Probability
{
    static class Episode24
    {
        static IDiscreteDistribution<bool> Flip() =>
            Bernoulli.Distribution(1, 1).Select(x => x == 0);

        public static void DoIt()
        {
            Console.WriteLine("Episode 24 - Gamblers Ruin Markov Chain generation");
            const int n = 5;

            var initial = Singleton<int>.Distribution(n);

            Func<int, IDistribution<int>> transition = x =>
              (x == 0 || x == 2 * n) ?
                Empty<int>.Distribution :
                Flip().Select(b => b ? x - 1 : x + 1);

            var markov = Markov<int>.Distribution(initial, transition);
            Console.WriteLine("Sample game");
            Console.WriteLine(markov.Sample().CommaSeparated());
            Console.WriteLine("Histogram of game lengths");
            Console.WriteLine(markov.Samples().Select(x => x.Count()).DiscreteHistogram());
        }
    }
}
```

In the next lesson, we will learn about the application of Markov Model.