An Introduction to Markov Chains

CSCI 2134 – Fall 2023
Alex Brandt

One of the best explanations of what a Markov chain is can be found at:
https://setosa.io/ev/markov-chains/

Reading through that link should be sufficient to understand what we need from Markov Chains. If you still want more explanation, or some explanation in the words of Alex, read on.

While the above link has some nice visuals, for our purposes we only need to know a very little bit about Markov chains. A Markov chain is made of two parts: a "current state" and a "transition matrix". At each "step", the Markov chain uses the transition matrix to move from the "current state" into a "next state", thus updating the current state.

The Transition Matrix

How does the matrix control transitions? The matrix encodes the probability of transition from one state to another. Each row of the matrix corresponds to one "from state". Each column of the matrix corresponds to one "to state".

Starting with index 0, the 0^{th} row of the matrix encodes all the probabilities of transitioning from the 0^{th} state to all of the other states. The (0,0) entry encodes the probability of moving from state 0 to state 0. The (0, 1) entry encodes the probability of moving from state 0 to state 1. The (i, j) entry encodes the probability of moving from state i to state j.

Example:

0.25	0.75
0.5	0.5

In this matrix, the probability of moving from state 0 to state 0 is 25%. The probability of moving from state 0 to state 1 is 75%. From state 1, there is equal probability of moving to either state 0 or state 1.

Notes:

1. The number of rows (and the number columns) encodes the number of states for the Markov chain.
2. The transition matrix must be square. Even if it "impossible" to move from state x to state y, we encode that by putting 0 as the matrix entry at position (x,y).
3. Each row of the transition matrix must sum to equal 1. The entries in a row correspond to the probabilities of transitioning from a particular state. Those probabilities have to sum to equal 1! We can't have probabilities over 100%, right?

Computing the "next state"

Okay, we have a transition matrix now. How do we actually use probabilities to make the transition happen?

Recall that we can generate random numbers using java.util.Random.nextDouble() or java.lang.Math.random() to get a random number uniformly distributed in the range [0,1). That is, a random floating point number x where 0 <= x < 1. We use that random number to choose the next state.

The algorithm is simple:

1. i = current state
2. r = random number
3. v = 0
4. for (j = 0; j < number of states; ++j) {
    a. v += transitionMatrix[i][j]
    b. if (r < v) { currentState = j; return j; }

Example:

Let the transition matrix be:

0.2	0.5	0.3
0.5	0.1	0.4
0.5	0.5	0.0

Let's run the above algorithm where the current state is 0. There is a 20% chance for the next state to 0 again. There is a 50% chance for the next state to be 1. There is a 30% chance for the next state to be 2.

We are going to partition the interval [0,1] into 3 parts, where the length of each partition corresponds to a next state, and the length of each partition corresponds to the probability of going to that state. For this matrix, [0, 0.2) corresponds to state 0. [0.2, 0.7) corresponds to state 1. [0.7, 1.0] corresponds to state 2.

We pick a random number between 0 and 1. Let's say we get r = 0.72. Since r falls into the interval [0.7, 1.0], the next state is 2.

In the above algorithm, the intervals are implicit. Consider the first loop. After executing a. v is equal to 0.2. If r < 0.2 the r must be in the interval [0, 0.2). Consider the next loop. After executing a. for the second time, v is equal to 0.7. If r < 0.7, then r must be in the interval [0.2, 0.7). Why? Because we already know from the previous loop that r is **not** less than 0.2.

Still have questions?

Post questions in our MS Teams in the "Assignment Discussion" channel.