Markov Chains, and the United Kingdom's Weather

Aniketh Kopalle

I have someone I'd like you to meet. His name is Andrey Markov, and he's the most badass, absolute mad rocker you'll ever meet. Bold, stubborn, idiosyncratic and rebellious, "Andrew the Furious" took up both math and political activism, writing a limerick "unfit for ladies' ears" when one worthy of his enmity joined the Academy of Sciences. He also, with considerable sincerity and politeness, requested to be excommunicated from the church as a reaction to Leo Tolstoy's notably un-requested excommunication. Today, though, we know him not for his admittedly fantastic facial hair (Figure 1 for reference), but for the Markov chain.

Figure 1. Truly fantastic facial hair indeed.

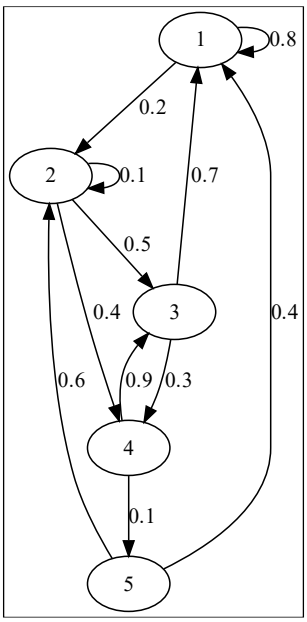

Figure 2, A Markov chain.

Markov pioneered Markov chains in the most unlikely of contexts, to analyse poetry. Analysing Alexander Pushkin's poem 'Eugene Onegin', he used his tool to calculate the distribution of vowels and consonants. Today, we know it as a major player in machine learning algorithms, like the ones used by billion-dollar firms to predict the movements of the stock market, and in PageRank, the first iteration of Google's search algorithm.

Markov chains represent states (nodes), and their transitions (edges), in a probabilistic sense. Each edge has a weight $0 \leq \omega \leq 1$, which represents the directed or undirected probability of transitioning between the two states it connects.

The power of Markov chains is that we can use techniques from linear algebra on a graph's adjacency matrix. Take Figure 2, a graph with five nodes, and its adjacency matrix M (the n^{th} row refers to all outgoing edges from node n, and the n^{th} column refers to all incoming edges to node n).

$$M = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.5 & 0.4 & 0 \\ 0.7 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.9 & 0 & 0.1 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix}$$

There's something very special about the matrix here. All of the components, row-wise, add up to one. It's a necessity, actually, for a Markov chain (you could interpret A such that the *columns* add to one, but that's not the norm).

Why do they have to add up to one? Well, at any given state, I need to either stay there, or I need to transition to a different state, with probability 1. The total probability of all possible outcomes from that state must be 1 - because I can't just vanish into a non-state. Put more succinctly, a row represents the probability distribution over possible next states, which always adds to 1.

You can do cool things with a transition matrix.

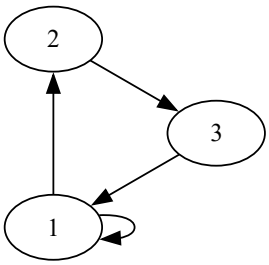

Figure 3. More manageable, right?

Did you know that given an adjacency matrix of *any* graph, A, A^2 counts the number of paths from any node to any other node, of length 2? In general, A^n counts the number of paths from any node to any other node of length n. The intuition is as follows, using a different, more manageable graph as a canvas (Figure 3.).

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

How do we calculate A^2?

$$A^2 = \begin{bmatrix} <1,1,1> \cdot <1,0,1> & <1,1,1> \cdot <1,0,0> & <1,1,1> \cdot <1,1,0> \\ <0,0,1> \cdot <1,0,1> & <0,0,1> \cdot <1,0,0> & <0,0,1> \cdot <1,1,0> \\ <1,0,0> \cdot <1,0,1> & <1,0,0> \cdot <1,0,0> & <1,0,0> \cdot <1,1,0> \end{bmatrix}$$

$$= \begin{matrix} 2 & 1 & 2 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

Let i, j be the row and column of a matrix.

Another way to represent matrix multiplication is like so: $[A^2]_{i,j} = \sum_{k=1}^{3} A_{i,k} A_{k,j}$, 3 because it's the number of columns of A.

If we look at the right-hand side, this is counting a potential path from i to k, and then multiplying it by another potential path from k to j. Multiplying because the entry $A_{i,k}A_{k,j}$ exists if and only if both of them are 1 (for matrix with 0 or 1 components, we'll get to transition matrices later). We're treating k as an "intermediate" node, counting any and all paths from i to *all* possible k to j, since we're iterating k from 1 to 3.

This isn't too rigorous, but the intuition holds, not just for A^2 but A^n in general, as any n-path from two nodes can be extended by multiplying by A again, resulting in a matrix of all $(n+1)$-paths.

This technique also works for matrices not just consisting of 0's and 1's! In fact, we can see the previous case as a *subset* of using real numbers between 0 and 1 inclusive! This is because we can interpret a 0 and a 1 as a 0% chance of being chosen, and a 100% chance of being chosen, meaning we could theoretically 0.31415 if we wanted to!

If we let our stochastic matrix M represent say, the weather, where each state is a sunny, snowy, etc. day, we could end up with something like Figure 4.

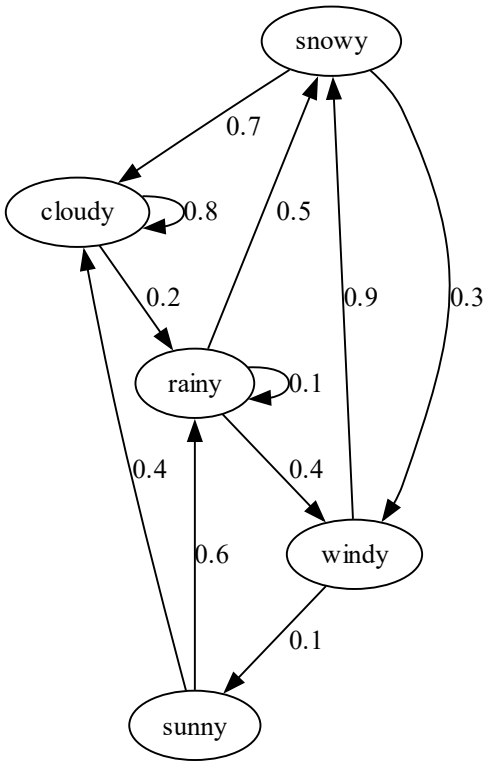

Figure 4. A surprisingly accurate weather system.

This is our adjacency matrix M with labels:

$$M = \begin{matrix} cloudy \\ rainy \\ snowy \\ windy \\ sunny \end{matrix} \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.5 & 0.4 & 0 \\ 0.7 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.9 & 0 & 0.1 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix}$$

Given what we've found out about matrices, we can answer cool and not-insignificant questions like 'what's the distribution of weather after 3 days given today's sunny?'. We just need to find M^3.

$$M^3 = \begin{bmatrix} 0.582 & 0.146 & 0.162 & 0.102 & 0.008 \\ 0.583 & 0.095 & 0.176 & 0.127 & 0.019 \\ 0.649 & 0.144 & 0.07 & 0.137 & 0 \\ 0.536 & 0.14 & 0.273 & 0.024 & 0.027 \\ 0.466 & 0.078 & 0.286 & 0.146 & 0.024 \end{bmatrix}$$

Since we've started at 'sunny', we can see that it's exceedingly likely to be 'cloudy' ($M_{5,1} = 0.466$), with a chance of ~~meatballs~~ 'snowy' ($M_{5,3} = 0.466$). Now I'm not too sure what country has weather like this, but it can't be too far off from the UK's weather.

But things can get a little tricky if we ask a question that doesn't mention timeframes. 'What's the distribution of weather in general?' is difficult to answer by just raising our matrix to powers!

We're going to have to find what's called the stationary distribution of our matrix.

Let's fix a starting point. Let's say I'm starting at node 1. I'm going to be referring to the nodes using numbering, just so that we can clearly see which is which in case we can't remember the ordering of our weather states. I'm going to represent that using a vector $\rho_1 = [1 \quad 0 \quad 0 \quad 0 \quad 0]$, which is saying that *the probability* of me being at node 1 is 1, node 2 is 0 and node 3 is 0. In case you're curious, the reason I've called the vector ρ_1 is because it's our starting distribution, i.e., our 0th distribution.

Let's multiply ρ_1 with $M = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.5 & 0.4 & 0 \\ 0.7 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.9 & 0 & 0.1 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix}$, to get ρ_2.

Why do we get ρ_2, you may ask? Well, it's really similar logic to raising M to a power. ρ_1 is $[1 \quad 0 \quad 0 \quad 0 \quad 0]$. We're taking a dot product of ρ_1 and the columns of M, meaning *given* we have a 100% chance of starting at node 1, we then 'branch out' (make a move) to see where we could travel from node 1. We can clearly see that we could either stay at node 1 with a probability of 0.8 or move to node 2 with a probability of 0.2.

$$\rho_1 M = [1 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.5 & 0.4 & 0 \\ 0.7 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.9 & 0 & 0.1 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix} = [0.8 \quad 0.2 \quad 0 \quad 0 \quad 0] = \rho_2$$

I'll do it again.

$$\rho_2 M = [0.8 \quad 0.2 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.5 & 0.4 & 0 \\ 0.7 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.9 & 0 & 0.1 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix} = [0.64 \quad 0.18 \quad 0.1 \quad 0.08 \quad 0]$$

Similar to last time, we're saying that *given an 80% chance of being at node 1, and a 20% chance of being at node 2, here's our new distribution after a second move.*

Let's use a little Python to do this many, *many* more times.

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from sympy.matrices import Matrix
>>> compose = lambda starting_distribution, adjacency_matrix, times: starting_distribution*(adjacency_matrix**times)
>>>
>>> a = Matrix([[1, 0, 0, 0, 0]])
>>> b = Matrix([[0.8, 0.2, 0, 0, 0], [0, 0.1, 0.5, 0.4, 0], [0.7, 0, 0, 0.3, 0], [0, 0, 0.9, 0, 0.1], [0.4, 0.6, 0, 0, 0]])
>>>
>>> print(compose(a, b, 4))
Matrix([[0.582200000000000, 0.135800000000000, 0.164800000000000, 0.107000000000000, 0.0102000000000000]])
>>> print(compose(a, b, 40))
Matrix([[0.587017873941750, 0.137347130761986, 0.161806208842766, 0.103480714957749, 0.0103480714957511]])
>>> print(compose(a, b, 400))
Matrix([[0.587017873941684, 0.137347130761997, 0.161806208842900, 0.103480714957669, 0.0103480714957669]])
>>>
```

When we use the `compose` function 4 times, we get a vector $\rho_5 = [0.5822, 0.1358, 0.1648, 0.107, 0.0102]$. When we use `compose` 40 times, we get something very similar ($\rho_6 = [0.5870, 0.1375, 0.1618, 0.1035, 0.0103]$), and when we use it *400* times, we get something is practically identical to the previous one. We can see that it's converging. In other words, when we started at node 1, as we keep moving $n \to \infty$ times, we converge to a vector! This is the stationary distribution of the matrix!

But there's a really interesting question. Does our original choice of starting node matter? Let's see.

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from sympy.matrices import Matrix
>>> compose = lambda starting_distribution, adjacency_matrix, times: starting_distribution*(adjacency_matrix**times)
>>>
>>> a = Matrix([[0, 0, 0, 0, 1]])
>>> b = Matrix([[0.8, 0.2, 0, 0, 0], [0, 0.1, 0.5, 0.4, 0], [0.7, 0, 0, 0.3, 0], [0, 0, 0.9, 0, 0.1], [0.4, 0.6, 0, 0, 0]])
>>>
>>> print(compose(a, b, 4))
Matrix([[0.582600000000000, 0.115400000000000, 0.170400000000000, 0.117000000000000, 0.0146000000000000]])
>>> print(compose(a, b, 40))
Matrix([[0.587017873942183, 0.137347130761935, 0.161806208841998, 0.103480714958226, 0.0103480714956604]])
>>> print(compose(a, b, 400))
Matrix([[0.587017873941684, 0.137347130761997, 0.161806208842900, 0.103480714957669, 0.0103480714957669]])
>>>
```

Here, I chose node 5 as my starting point, travelling everywhere starting at node 5. Lo and behold, we end up with the *same matrix!* This is terrific! Here's another one to drive the point in!

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from sympy.matrices import Matrix
>>> compose = lambda starting_distribution, adjacency_matrix, times: starting_distribution*(adjacency_matrix**times)
>>>
>>> a = Matrix([[0.1, 0.2, 0.3, 0.15, 0.25]])
>>> b = Matrix([[0.8, 0.2, 0, 0, 0], [0, 0.1, 0.5, 0.4, 0], [0.7, 0, 0, 0.3, 0], [0, 0, 0.9, 0, 0.1], [0.4, 0.6, 0, 0, 0]])
>>>
>>> print(compose(a, b, 4))
Matrix([[0.588375000000000, 0.133800000000000, 0.163770000000000, 0.102375000000000, 0.0116800000000000]])
>>> print(compose(a, b, 40))
Matrix([[0.587017873941615, 0.137347130762002, 0.161806208843005, 0.103480714957600, 0.0103480714957794]])
>>> print(compose(a, b, 400))
Matrix([[0.587017873941684, 0.137347130761997, 0.161806208842900, 0.103480714957669, 0.0103480714957669]])
>>>
```

Here I chose a really weird ρ_1, with a 10% chance of starting at node 1, 20% chance of starting at node 2, etc., but *we still got the same value!*

Mathematically, $v \lim_{n\to\infty} M^n = \rho$, where v is any vector (whose components add to 1) ρ is our stationary distribution! If you know a little bit of linear algebra, you'll know that this is quite similar to the eigenvector of M. I'd just like to add that this only works for *some* Markov chains (boo, I know!). They need to be what's called **irreducible** and **aperiodic**, which ours happens to be! Loosely speaking, they assert that a unique stationary distribution exists if and only if the Markov chain is connected, i.e., there exists a path from every node to every other node, and if there 'isn't a pattern' to when we reach a state i.

Back to the stationary distribution, guess what happens when we solve this equation:

$$\rho M = \rho$$

$$\rho = [a \quad b \quad c \quad d \quad e]$$

$$[a \quad b \quad c \quad d \quad e]\begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.5 & 0.4 & 0 \\ 0.7 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0.9 & 0 & 0.1 \\ 0.4 & 0.6 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.8a + 0.7c + 0.4e \\ 0.2a + 0.1b + 0.6e \\ 0.5b + 0.9d \\ 0.4b + 0.3c \\ 0.1d \end{bmatrix}^T$$

$$\begin{bmatrix} 0.8a + 0.7c + 0.4e \\ 0.2a + 0.1b + 0.6e \\ 0.5b + 0.9d \\ 0.4b + 0.3c \\ 0.1d \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix}$$

We can turn this to a "cleaner" equation like so:

$$\begin{bmatrix} -0.2a + 0.7c + 0.4e \\ 0.2a - 0.9b + 0.6e \\ 0.5b - c + 0.9d \\ 0.4b + 0.3c - d \\ 0.1d - e \end{bmatrix} = \vec{\mathbf{0}}$$

We can combine all of this into an augmented matrix and solve using Gauss-Jordan elimination.

$$\left[\begin{array}{ccccc|c} -0.2 & 0 & 0.7 & 0 & 0.4 & 0 \\ 0.2 & -0.9 & 0 & 0 & 0.6 & 0 \\ 0 & 0.5 & -1 & 0.9 & 0 & 0 \\ 0 & 0.4 & 0.3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & -1 & 0 \end{array}\right]$$

Which if we solve, we get

$$a = \frac{624}{1063} \approx 0.5780$$
$$b = \frac{146}{1063} \approx 0.1373$$
$$c = \frac{172}{1063} \approx 0.1618$$
$$d = \frac{110}{1063} \approx 0.1035$$
$$e = \frac{11}{1063} \approx 0.0103$$

Which is the same as our 'simulation'! We could have used the characteristic polynomial of M, but in my opinion it can get quite messy.

So, what does this mean? Diving into our weather system, on any given day, it's exceedingly likely for it to be cloudy, just under 60% of the time! It also means that it's sunny *just over 1% of the time!* That, I think, is pretty strong evidence that this is based off the UK's weather… perhaps excluding the snow.

But that raises an important question. Why is it that solving $\rho M = \rho$ results in the *same* ρ as $\hat{v} \lim_{n\to\infty} M^n = \rho$, where $\hat{v}$ is any normalised vector. Without getting too much into the complexities of dominant eigen-things, think about what it is that is being done in $v \lim_{n\to\infty} M^n = \rho$. To make it easier, let's use recurrence relations that we discussed previously like so:

$$\rho_{n+1} = \rho_n M$$

Intuitively, we can see that in order for $\rho = \rho_\infty$ to exist, i.e., for our stationary distribution to exist, our changes must get smaller and smaller when we multiply by M. From this, it's intuitively clear that we are indeed looking for $\rho M = \rho$, as what change could be 'smaller' than, well, no change at all?

Obviously, this isn't rigorous (not at all). But it makes sense.

What I've shown is likely less than 1% of the incredibly, *incredibly* cool things you can do with them. Going along with the weather example, we could calculate how long it might take you to get back to a sunny day, given today's sunny, etc. (mean recurrence time). Or we could even quantify how 'important' today's weather is in predicting future weather (it's absolutely *insane* that we can do this!) via mixing times, a way of quantifying how

many steps (how large n has to be) it takes before our $\hat{v}M^n$ starts looking like our stationary distribution.

Just the sheer cleverness of some of these ideas, and the fact that they are used to model *everything* makes them undoubtedly my favourite area of math. They're abstract enough to capture the essence of any random process, but concrete enough to explain to just about anyone, and to apply them on just about anything!

References

Basharin, G., Langville, A. and Naumov, V. (2004). The life and work of A.A. Markov. *Semantic Scholar*, [online] 386. doi:https://doi.org/10.1016/J.LAA.2003.12.041.

O'Connor, J.J. and Robertson, E.F. (n.d.). *Andrei Andreyevich Markov*. [online] MacTutor Index. Available at: https://mathshistory.st-andrews.ac.uk/Biographies/Markov/.

Jauregui, J. (2012). *Markov chains: examples Markov chains: theory Google's PageRank algorithm Math 312*. [online] Available at: https://www2.math.upenn.edu/~kazdan/312F12/JJ/MarkovChains/markov_google.pdf.

Aldridge, M. (n.d.). *Section 10 Stationary distributions | MATH2750 Introduction to Markov Processes*. [online] *mpaldridge.github.io*. Available at: https://mpaldridge.github.io/math2750/S10-stationary-distributions.html.

Valiant, G. and Wootters, M. (2025). *CS265/CME309: Randomized Algorithms and Probabilistic Analysis Lecture #15: Mixing Times, Strong Stationary Times, and Coupling*. [online] Available at: https://web.stanford.edu/class/cs265/Lectures/Lecture15/l15.pdf