Numerical Methods

Math 3338 – Spring 2022

Worksheet 17

Eigenvector Applications - Markov Chains

1 Reading

CP
NMEP

Table 1: Sections Covered

2 What is a Markov Chain

Think about a system with a sequence of states s_n so that the probability of s_n depends only on s_{n-1}. A classic example is a simple board game like Chutes and Ladders, where you land at the end of the turn depends only on where you are now.

Let's do an example. Figure 1 shows a very simple game. If you are standing on A, you have a .15

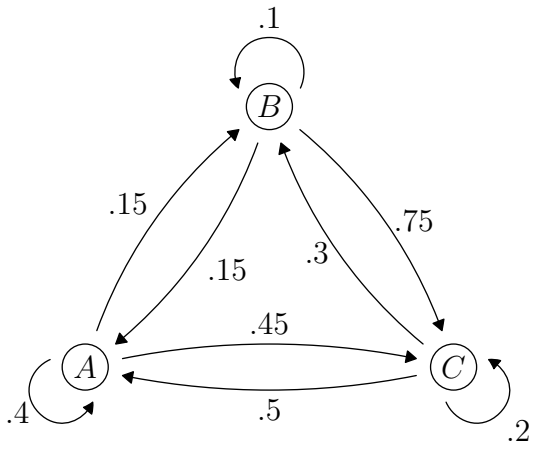

Figure 1: A simple game.

probability of moving to B, .45 of moving to C and a .4 of not moving at all. Using this we can create a matrix.

$$M = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} .4 & .15 & .45 \\ .15 & .1 & .75 \\ .5 & .3 & .2 \end{bmatrix} \end{array} \tag{1}$$

The matrix M is the transition matrix of Figure 1. Notice the sum of each row is 1, this should tell you have the matrix is arranged.

Let our sequence of states be denoted by x_n and say we start on A, then $x_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$. We want to know the probability of being on each vertex after 1 'round'. We should be able to use logic to determine[1]

[1] If this isn't clear, think about the probability of going to each space starting on A.

that $x_1 = \begin{bmatrix} .4 \\ .15 \\ .45 \end{bmatrix}$. Of course, to find x_2 it becomes much more difficult. This is where the Markov matrix becomes useful, notice $x_1 = Ax_0$. Expanding on this we see,

$$\begin{aligned} x_n =& Ax_{n-1} \\ =& AAx_{n-2} \\ =& A^2x_{n-2} \\ =& \dots \\ =& A^nx_0 \end{aligned}$$

If we let $x = \lim x_n$ then $x = Ax$, which implies that x is an eigenvector of A and 1 is an eigenvalue[2]. This is actually a little hand wavy at the moment. In reality, we want to find A^n. Think about diagonalizing A,

$$P^{-1}AP = D$$

Recall from linear algebra that P is the matrix of eigenvectors of A and D is the diagonal matrix with eigenvalues on the diagonal. Then $A^n = (PDP^{-1})^n = PD^nP^{-1}$. The eigenvalues of a markov matrix all satisfy $|\lambda| \leq 1$ with exactly one equal to 1. This implies D^n converges to a matrix with exactly one 1 on the diagonal and 0's everywhere else.

The steady state matrix $A_{steady} = P \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} P^{-1}$ is the ultimate convergence of the system. In otherwords, if you played an infinite number of rounds, the entries of A_{steady} would be the probability you'd be on any space given a starting point.

[2]Assuming that the sequence converges, which it turns out always does for Markov matrices.

Numerical Methods

Math 3338 – Spring 2022

Homework 17 (Due: Tuesday, March 22)

Problem 1 (1 pt) Create a function `steady_state(A)` that returns the steady state of a Markov matrix A. You can always test your answer by choosing a matrix A and taking repeated powers; however, your function should be smarter than that.

For Problems 2 – 4 use the board in Figure 1.

Problem 2 (1 pt) If you start on A, what is the probability you land on A again after

1. 1 round
2. 10 rounds
3. 100 rounds
4. 1000 rounds

Problem 3 (1 pt) Find the steady state of the board. What is the probability you are on A in the steady state if you start on A?

Problem 4 (1 pt) This is an optional, but highly recommended, problem. Create a simulation that will play `N` rounds on the board starting on A. What do I mean? You start on A and the next turn you have a .15 probability of moving to B. Do not use a Markov matrix to solve this problem, you want explicit states, not probabilities. Create a function called `play_round(B,location)` where `B` is the board and `location` is your current location. This will return the next location. Next create a function `play_game(B,N)` where B is the board and N is the number of rounds. Use these functions to simulate 1000 games of 100 rounds each.

Find the distribution of your location after each of the 100 rounds (i.e. add all the states and divide by 1000). Compare this to $A^{100}v$ where v is the vector corresponding to starting on A. How close are your answers?

Problem 5 (1 pt) Let's play Chutes and Ladders see Figure 2 for the board. The way the game works is you spin a dial and move 1, 2 or 3 spaces. If you land at the bottom of a ladder, you go to the connected space. Top of a slide? You're going backwards. For example, if you land on space 3, you advance to space 10. If you land on space 42[3], you win!

1. Create the Markov matrix for this board. This will be a large, sparse matrix. Try to be clever about how you represent it and enter it into Python (you'll eventually want to turn it into a numpy matrix).
2. How many rounds do you need to play before the probability of winning is greater than 50%?
3. What does the steady state matrix look like for this game? Explain why.
4. Create a function `heatmap(B,n)` that creates a heatmap of likely locations in round n. The function `matplotlib.pyplot.imshow` will probably be useful as will reshaping arrays using numpy.

[3]You need to land exactly on the space, so if you roll a 3 on space 40 you stay on space 40.

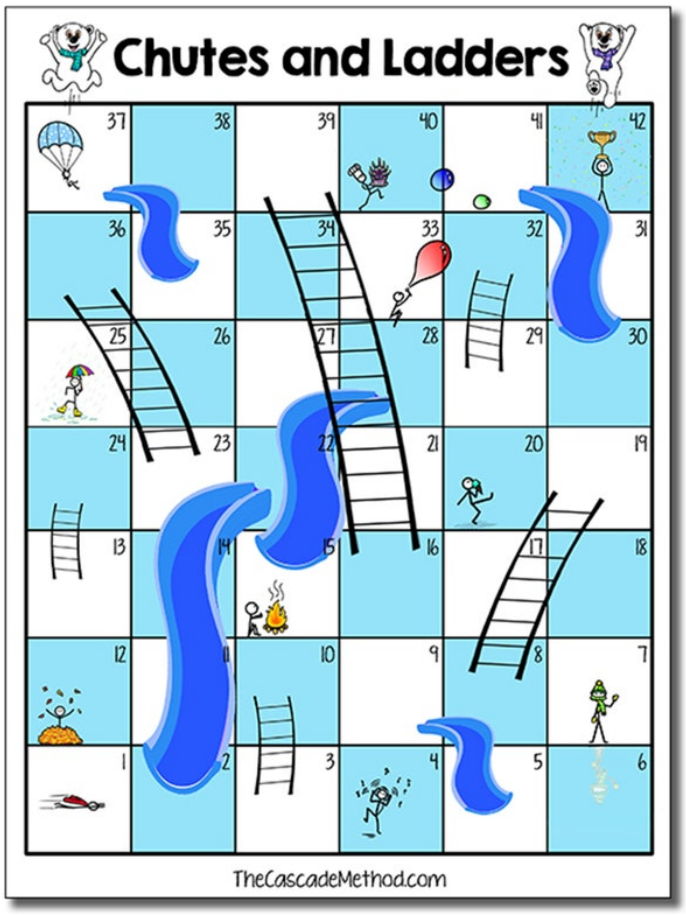

Figure 2: Chutes and Ladders