5.2 MARKOV CHAINS II

1. Ergodic Markov Chains

Recall that a chain is called ergodic if there is a path from any vertex to any other vertex in the chain. For instance, the chain

$$\begin{tikzcd}
& 1 \arrow[d, bend right, "1"'] & \\
3 \arrow[r, bend right, "1"'] & 2 \arrow[l, bend right, "\frac{1}{3}"'] \arrow[u, bend right, "\frac{1}{3}"'] \arrow[r, bend right, "\frac{1}{3}"'] & 4 \arrow[l, bend right, "1"']
\end{tikzcd}$$

with corresponding matrix

$$P = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \end{pmatrix}$$

is ergodic.

Associated to any ergodic chain we have a unique column vector v with the properties

1. $P \cdot v = v$
2. $\sum_i v[i] = 1$
3. $v[i] \geq 0$ for all i

The three properties above characterise the vector v uniquely, and allow us to compute it (Item 1. and 2. gives enough equations to solve for v).

In the case that all state changes going out of a vertex have equal probability, the probability becomes a lot easier. In the example above we do the following:

There is 1 node pointing to the node 1,

There are 3 nodes pointing to the node 2.

There is 1 node pointing to the node 3.

There is 1 node pointing to the node 4.

The sum of these numbers is $1 + 3 + 1 + 1 = 6$ which gives us the probability vector

$$v = \frac{1}{6}\begin{pmatrix} 1 & 3 & 1 & 1 \end{pmatrix}.$$

Let W be the square matrix with columns equal to v. This matrix can also be computed as

$$W = lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} P^i$$

and so $W[i][j]$ (which is equal to $v[i]$ is the proportion of turns we will be at state i having started at state j. This number is independent of the starting vertex j in any ergodic chain. The number $\frac{1}{v[i]}$ computes the expected number of turns needed between leaving state i until returning to i.

The **fundamental matrix** of an ergodic chain is

$$Z = (I - P + W)^{-1}.$$

Note that we use the name "fundamental matrix" for both absorbing chains an ergodic chains, but that these matrices are defined differently for these chains.

The expected number of turns needed to move from state i to state j is the number

$$M[i][j] = \frac{Z[i][i] - Z[i][j]}{v[i]}.$$

We see that $M[i][i] = 0$ which tells us that zero moves are needed to move from state i to state i. If we want to calculate the expected number of turns needed to move from state i to state i, where we must leave state i, then the number we want is $\frac{1}{v[i]}$.

Note that calculating expected number of turns of certain events only gives you information about what happens on average with many tries. The variance, which gives information about the expected distance from the average, is also important. For computing variance, and many other interesting quantities, you can have a look at Fronter and the links listed there.

Problems:

Exercise. 1. a) *Give an example of a Markov chain which does not have any absorbing states and is not ergodic.*

b) *Give an example of a Markov chain which has an absorbing state and is ergodic.*

Exercise. 2. *You will need to use a computer for the next problem. Design a maze on at least a 4×4 grid (you decide which moves are allowed, but the resulting Markov Chain should be ergodic).*

a) *Using the methods described above, calculate the expected number of turns needed to move from the upper left corner to the lower right corner.*
b) *Run a simulation to find out if your calculation in a) is correct.*

Answers/Solutions:

Exercise 1.

a) $$\begin{tikzcd} 1 \arrow[r, "1"] & 2 \arrow[r, bend right, "1"'] & 3 \arrow[l, bend right, "1"'] \end{tikzcd}$$

It has no absorbing states, as there are no loops. Also, it is not possible to move from 2 to 1, so the chain is not ergodic.

b) The Markov chain given by one state and one loop with probability 1.

Exercise 2.

I use a 4×4 grid with states numbered as follows.

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix}$$

The markov chain has 16 states and so we will need to work with 16×16 matrices. Obviously we will be using a computer. I use Python with numpy for this problem.

I first create the matrix of a graph with 16 vertices to encode the maze. I add a path through the maze to ensure connectedness, and then open a few more doors.

I then rescale the entries in the matrix so that it is a Markov Chain

```
N = np.zeros((16))
createMarkov(P,N)
n = np.sum(N) # Twice the number of doors\\
```

At the same time I calculate twice the number of doors and the number of doors at each state. This number will be needed in the computation of the probability vector v. The rescaling is implemented as a function which works for Markov chains with more or less states than in this example.

I then create the fundamental matrix Z

```
w = np.array([x/n for x in N])
W = np.transpose(np.array([w for x in range(16)]))
Z = np.linalg.inv(np.identity(16)-P+W)
```

and read of that the expected number of turns is about 106.

b)

The simulation is set up differently. I think of my maze as a list of lists, where the list at position i is the set of neighbouring states to state i. I create this list from the Markov Chain I constructed earlier.

```
L = []
for i in range(16):
    l = []
    for j in range(16):
        if P[j][i] != 0:
            l.append(j)
    L.append(l)
```

This setup allows me write a fairly simple code for the simulation

```
seed()
count = 0.0
numsim = 100000
i = 0
while i < numsim:
    state = 0
    while state != 15:
        state = L[state][randint(0,len(L[state])-1)]
        count += 1
    i += 1
```

I run the simulation 10000 times and get that the expected number of turns is about 107.

The full source code is included on fronter.