Example sheet 4

Random processes
Data Science—DJW—2021/2022

Question 1. For the Cambridge weather simulator, example 10.1.1 in lecture notes, show that

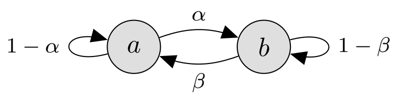
$$\mathbb{P}(X_3 = r \mid X_0 = g) = \sum_{x_1,x_2} P_{gx_1} P_{x_1x_2} P_{x_2r}.$$

Explain your reasoning carefully.

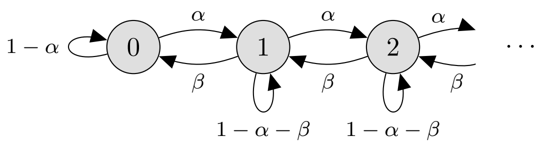
Question 2. Draw the state space diagram for this Markov chain.

```
def rw():
    MAX_STATE = 9
    x = 0
    while True:
        yield x
        d = numpy.random.choice([-1,0,1], p=[1/4,1/2,1/4])
        x = min(MAX_STATE, max(0, x + d))
```

Question 3. Here is the state space diagram for a Markov chain. Find a stationary distribution. Is it unique?



Question 4. Here is the state space diagram for a Markov chain, with state space $\{0, 1, 2, \dots\}$. It is parameterized by α and β, with $0 < \alpha < \beta$ and $\alpha + \beta < 1$. Let $\pi_n = (1 - \alpha/\beta)(\alpha/\beta)^n$, $n \geq 0$. Show that π is a stationary distribution.



Question 5. Let $X_n \in \mathbb{N}$ be the number of infectious people on day n of an epidemic, and consider the Markov chain model

$$X_{n+1} = X_n + \text{Poisson}(rX_n/d) - \text{Bin}(X_n, 1/d).$$

We would like to compute the probability that, starting from state $X_0 = x$, the epidemic dies out i.e. hits state 0. In order to solve this by computer, we'll cut the state space down to $\{0, \dots, N\}$, for some sufficiently large N, by amalgamating all the states with $\geq N$ infected and letting the transition from state N back to itself have probability 1.

(a) Give pseudocode to compute the transition matrix. You should give your answer in terms of `binom.pmf` and `poisson.pmf`, the likelihood functions for the two distributions in question.

(b) Give pseudocode to compute the probability that the epidemic dies out, starting from any initial state $x \in \{0, \dots, N\}$. (For $r = 1.1$ and $d = 14$, for $X_0 = 50$, the probability is 0.7%.)

Question 6. We're given a connected, undirected graph. Consider a random walk on the vertices of this graph, as follows: each timestep it takes one of the edges chosen at random, each edge from its current vertex equally likely. Find the stationary distribution using detailed balance.

Question 7. Let X_n be a mean-reverting random walk,

$$X_{n+1} = \mu + \lambda(X_n - \mu) + N(0, \sigma^2) \quad \text{where} \quad -1 < \lambda < 1.$$

The stationary distribution for this process is a Normal distribution. Find its parameters.

Question 8. The Internet uses an algorithm called TCP to manage congestion. For every data flow, the sender maintains a congestion window $W_n \in \mathbb{R}$. It keeps roughly W_n packets in flight, thus the transmission rate is $W_n/$`RTT` packets per second where `RTT` is the round trip time ('ping latency') between sender and receiver. The sender updates W_n every time it receives an acknowledgement of a packet, by

$$W_{n+1} = \begin{cases} W_n + 1/W_n & \text{if the packet didn't experience congestion} \\ W_n/2 & \text{if the packet did experience congestion.} \end{cases}$$

Suppose that packets experience congestion with probability p, independently. Write down a drift model for W_n and find the fixed point.

Question 9. Consider a moving object with noisy location readings. Let X_n be the location at timestep $n \geq 0$, and Y_n the reading. Here's the simulator.

```python
def hmm():
    MAX_STATE = 9
    x = numpy.random.randint(low=0, high=MAX_STATE+1)  # initial location X_0
    while True:
        e = numpy.random.choice([-1,0,1])
        y = min(MAX_STATE, max(0, x + e))  # noisy reading of location
        yield y
        d = numpy.random.choice([-1,0,1], p=[1/4,1/2,1/4])
        x = min(MAX_STATE, max(0, x + d))  # new location at next timestep
```

We'd like to infer the location X_n, given readings $y_0, \ldots, y_n$.

(a) Give justifications for the following three equations, which give an inductive solution. First the base case,

$$\Pr(x_0 \mid y_0) = \text{const} \times \Pr(x_0) \Pr(y_0 \mid x_0),$$

and next two equations for the induction step,

$$\Pr(x_n \mid h) = \sum_{x_{n-1}} \Pr(x_{n-1} \mid h) \Pr(x_n \mid x_{n-1})$$

$$\Pr(x_n \mid h, y_n) = \text{const} \times \Pr(x_n \mid h) \Pr(y_n \mid x_n).$$

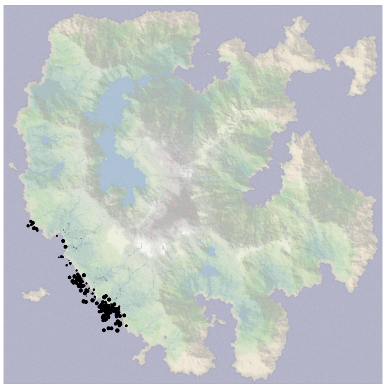
In these two equations, h stands for $(y_0, \ldots, y_{n-1})$, and we'll assume we've already found $\Pr(x_{n-1}|h)$.

(b) Give pseudocode for a function that takes as input a list of readings $(y_0, \ldots, y_n)$ and outputs the probability vector

$$\left[\pi_0, \ldots, \pi_{\texttt{MAX\_STATE}}\right], \qquad \pi_x = \mathbb{P}(X_n = x \mid y_0, \ldots, y_n).$$

(c) If your code is given the input $(3, 3, 4, 9)$, it should fail with a divide-by-zero error. Give an interpretation of this failure.

Question 10 (Stoat-finding challenge, not for supervision). Work through the notebook at `https://www.cl.cam.ac.uk/teaching/2122/DataSci/datasci/ex/ex4.html`, in which you will implement a 'particle filter' to solve question 9 on a larger map, using computational methods rather than exact calculations. Submit your answer on Moodle.



Hints and comments

Question 1. In section 11.1 we calculated $\mathbb{P}(X_2 = r \,|\, X_0 = g)$, using two tools: the memoryless property of Markov chains, and 'resetting the clock'. Use the same method here. Start by using the law of total probability, conditioning on X_1. Then condition on X_2. (Or you can do X_2 first, or you can do both together.)

Question 2. First identify the state space, i.e. the set of possible values for x. Looking at the code, we see that x can only ever be an integer in $\{0, 1, \ldots, 9\}$, so this is the state space. Next, draw arrows to indicate transitions between states. Make sure that at every node you draw, the probabilities on all outgoing edges sum up to one. You don't need to draw every state in your state space diagram: just show a typical state, and also the edge cases.

Question 3. The equations to solve are

$$\pi_x = \sum_y \pi_y P_{yx} \text{ for all } x \qquad \text{and} \qquad \sum_x \pi_x = 1.$$

In section 11.4 we derived these, but for answering questions you should just remember them and apply them. Also, in your aswer, you should mention *irreducibility*, as defined in section 11.4. You should end up with the answer

$$\pi_a = \frac{\beta}{\alpha+\beta} \qquad \pi_b = \frac{\alpha}{\alpha+\beta}.$$

Question 4. When a question asks "show that π is a stationary distribution", you don't have to set about trying to *solve* the equations

$$\pi_x = \sum_y \pi_y P_{yx} \quad \text{for all } x.$$

You just need to *verify* that the π you're given does indeed solve these equations. For a simple state space like the one in this question, for a given x, the transition probability P_{yx} is zero for most states y, so it's easy to verify. Write out one equation for π_0, and then write out another equation for π_x for a generic $x > 0$.

You should also show that π is indeed a distribution! In this question, to show that $\sum_{n=0}^{\infty} \pi_n = 1$, either remember the formula for a geometric series ($\sum_{n=0}^{\infty} ar^n = 1/(1-r)$ for $|r| < 1$), or spot that π_n is the p.m.f. for a Geometric distribution and must necessarily sum to 1. (There are two flavours of the Geometric distribution. Wikipedia lists both.)

Question 5. This is a hitting probability question, just like example 11.2.1 from lecture notes. The only tricky bit is working out the transition matrix. For the matrix

$$\mathbb{P}(X_{n+1} = j \mid X_n = i) = \mathbb{P}(j = i + I - R) = \mathbb{P}(I = R + j - i)$$

where I is the number of newly infected, and R is the number of recoveries. Use the law of total probability, conditioning on R.

You should implement this in Python. Unless you implement it yourself, and do the sanity check that the rows of your transition matrix sum to one, it's very easy to make a mistake! For $r = 1.1$ and $d = 14$, cutting the state space down to $\{0, \ldots, 200\}$ is sufficient.

Question 6. First write out the transition probability: $P_{xy} = 0$ if there's no $x \leftrightarrow y$ edge, and $P_{xy} = 1/n_x$ otherwise, where n_x is the number of edges incident at vertex x. In indicator notation, $P_{xy} = 1_{x \leftrightarrow y}/n_x$. Then, see if you can spot a simple solution to the detailed balance equation.

Question 7. The state space is $\mathbb{R}$ which is not countable, so all the sum-based equations from lectures don't work. We have to go right back to the definition of stationarity, and write it in a way that works for continuous random variables: *a distribution π is a stationary distribution if*

$$X_0 \sim \pi \quad \Longrightarrow \quad X_1 \sim \pi.$$

Review the calculations in section 11.4 where we derived $\pi = \pi P$ for discrete state-space Markov chains, and think: can I use similar reasoning to derive the parameters of the Normal distribution that the question tells us is stationary?

Question 8. See section 11.7. The calculation is very simple.

The average transmission rate is $\hat{w}/\mathsf{RTT}$ packets per second, where $\hat{w}$ is the fixed point you found. This should roughly agree with the famous TCP throughput equation

$$\text{throughput} = \frac{\text{packet size}\sqrt{3/2}}{\mathsf{RTT}\sqrt{p}} \text{ bytes/sec}$$

which is in widespread use in network engineering. This formula involves slightly different assumptions (it assumes periodic losses rather than random) so your answer won't be exactly the same.

Question 9. This is a hidden Markov model, as mentioned in section 10.4:

$$\begin{tikzcd}
X_0 \arrow[r] \arrow[d] & X_1 \arrow[r] \arrow[d] & X_2 \arrow[r] \arrow[d] & \cdots \\
Y_0 & Y_1 & Y_2 &
\end{tikzcd}$$

Part (a). The second equation requires the law of total probability (with baggage), and the third equation requires Bayes's rule (with baggage h). 'With baggage' is described in section 11.1. The idea of these manipulations is to put the probability expressions into a form where you can leverage memorylessness: "X_n is generated based *only* on X_{n-1}, and Y_n is generated based *only* on X_n".

Part (b). Let $\pi^{(n)}$ be the probability vector at timestep n. Compute $\pi^{(0)}$ from the first equation. Then, iteratively apply the next two equations, to compute $\pi^{(n)}$ from $\pi^{(n-1)}$. Your implementation should use two matrices, $P_{ij} = \mathbb{P}(X_n = j \mid X_{n-1} = i)$ and $Q_{xy} = \mathbb{P}(Y_n = y \mid X_n = x)$. The first is the transition matrix that we're used to from Markov Chains, and the second is called the emission matrix.

Supplementary questions

These questions are not intended for supervision (unless your supervisor directs you otherwise).

Question 11 (Google PageRank). Consider a directed acyclic graph representing the web, with one vertex per webpage, and an edge $v \to w$ if page v links to page w. Consider a random web surfer who goes from page to page according to the algorithm

```
d = 0.85
def next_page(v):
    neighbours = list of pages w such that v → w
    a = random.choice(['follow_link','teleport'], p=[d,1-d])
    if a=='follow_link' and len(neighbours) > 0:
        return random.choice(neighbours)
    else:
        V = list of all web pages
        return random.choice(V)
```

This defines a Markov chain. Explain why the chain is irreducible. Show that the stationary distribution π solves

$$\pi_v = \frac{1-d}{|V|} + d \sum_{u:u\to v} \frac{\pi_u}{|\Gamma_u|}$$

where $|V|$ is the total number of web pages in the graph, and $|\Gamma_u|$ is the number of outgoing edges from u.

Compute the stationary distribution for this random web surfer model, for the graph in lecture notes example 11.2.1. Repeat with $d = 0.05$. What do you expect as $d \to 0$? What do you expect if $d = 1$?

The equation for π_v defines a scaled version of PageRank, Google's original method for ranking websites.

Question 12 (Hitting times). Consider the random web surfer model, for the graph in lecture notes example 11.2.1. Let t_x be the expected time, starting from x, until the surfer reaches Twitter (page 5). Derive the equations

$$t_x = 1 + \sum_y P_{xy} t_y \text{ for all } x \neq 5, \quad t_5 = 0$$

and solve with numpy.

Expected answer: [6.67, 5.25, 6.17, 5.08, 6.08, 0].

Question 13. The code from question 9(c) can fail with a divide-by-zero error. This is undesirable in production code! One way to fix the problem is to modify the Markov model to include a 'random teleport'—to express the idea 'OK, our inference has gone wrong somewhere; let's allow our location estimate to reset itself'. We can achieve this mathematically with the following model: with probability $1-\varepsilon$ generate the next state as per line 9, otherwise pick the next state uniformly from $\{0, 1, \ldots, \texttt{MAX\_STATE}\}$. Modify your code from question (b) to reflect this new model, with $\varepsilon = 0.01$.

Alternatively, we could fix the problem by changing the model to express 'OK, this reading is glitchy; let's allow the code to discard an impossible reading'. How might you change the Markov model to achieve this?

Question 14. The Markov model for motion from question 2 is called a *simple random walk (with boundaries)*; it chooses a direction of travel independently at every timestep. This is not a good model for human movement, since people tend to head in the same direction for a while before changing direction.

(a) Let $V_n \in \{-1, 0, 1\}$ be a Markov chain: let $V_{n+1} = V_n$ with probability 0.9, and let V_{n+1} be chosen uniformly at random from $\{-1, 0, 1\}$ with probability 0.1. Draw a state space diagram for this Markov chain.

Interpret V_n as the velocity of our moving object at timestep n, and let $X_{n+1} = \max(0, \min(9, X_n + V_n))$.

(b) Draw the state space diagram for (X_n, V_n).

(c) Give pseudocode to compute the stationary distribution.