Example sheet 4

Random processes
Data Science—DJW—2023/2024

Questions labelled * are more challenging (but should still be attempted!). For some questions you can test your answers using the online tester; there is a notebook with templates for answers and instructions for submission on the course materials webpage. Following this example sheet is a page with hints for each question.

Question 1. 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 2. We're given a sequence $[x_0, x_1, \ldots, x_n]$, and we decide to model it as the Markov chain

$$X_i = \mu + \lambda(X_{i-1} - \mu) + N(0, \sigma^2).$$

(When $|\lambda| < 1$ this tends to fluctuate around μ, as illustrated in lecture notes example 11.1.6, so it's known as a "mean-reverting random walk".) Explain how to estimate λ, μ, and σ. *[Optional: To test your code using the online tester, fill in the answer template for* ***`fit_mrrw`****.]*

Question 3*. Consider a dataset of average November temperatures in Cambridge. Let $\texttt{Temp}_i$ be the temperature for record i in the dataset, $i = 1, \ldots, n$, and let t_i be the year. Assume the records are sorted in increasing order of year, and that there are no gaps. Consider two models for this dataset: the rich model

$$\texttt{Temp}_i = \alpha + \gamma(t_i - 2000) + \lambda\texttt{Temp}_{i-1} + N(0, \sigma^2)$$

and the simpler model

$$\texttt{Temp}_i = \alpha + \gamma(t_i - 2000) + N(0, \sigma^2)$$

(a) Explain how to fit both models.

(b) Explain how to test the hypothesis that the simpler model is adequate.

[Optional: To test your code using the online tester, fill in the answer templates for ***`fit_climat0`****,* ***`fit_climate1`****, and* ***`test_climate0`****.]*

Question 4. For the Cambridge weather simulator, example 11.1.1 in lecture notes, show that

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

Question 5*. Consider this code for generating random variables $X \to Y \to Z$:

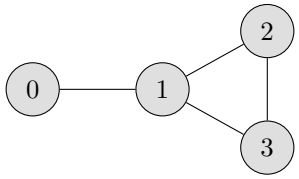
```
x = np.random.uniform()
y = np.random.binomial(n=1, p=x)
z = np.random.normal(loc=y, scale=ε)
```

Show that

$$\Pr_Y(1 \mid X = x, Z = z) = \frac{x}{x + (1-x)e^{(1-2z)/2\varepsilon^2}}.$$

How does $\Pr_Y(1 \mid X = x, Z = z)$ depend on x and z when $\varepsilon \approx 0$? What if ε is very large?

Question 6. Consider a random walk on the vertices of the undirected graph below, as follows: each timestep we take one of the edges chosen at random, each edge from our current vertex equally likely. Find the stationary distribution.



[Optional: To test your code using the online tester, fill in the answer template for `graphrw_stationary`*.]*

Question 7*. Let $X_0, X_1, X_2, \ldots$ be a mean-reverting random walk, i.e. a Markov chain

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

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

Question 8. 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.

```
1  def hmm():
2      MAX_STATE = 9
3      x = numpy.random.randint(low=0, high=MAX_STATE+1)  # initial location X_0
4      while True:
5          e = numpy.random.choice([-1,0,1])
6          y = min(MAX_STATE, max(0, x + e))  # noisy reading of location
7          yield y
8          d = numpy.random.choice([-1,0,1], p=[1/4,1/2,1/4])
9          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 a probability vector for the posterior distribution of X_n, in other words it returns $[\pi_0, \ldots, \pi_{\texttt{MAX\_STATE}}]$ where

$$\pi_x = \mathbb{P}(X_n = x \mid y_0, \ldots, y_n).$$

(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.

[Optional: To test your code using the online tester, fill in the answer template for `hmm_predict`*.]*

Question 9. The code from question 8 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?

Hints and comments

Question 1. 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 2. Follow the same pattern as example 12.1.1. You should conclude that you need to solve a least-squares optimization for the model

$$x_i \approx \mu + \lambda(x_{i-1} - \mu), \quad 1 \leq i \leq n.$$

This isn't a proper linear model, because proper linear models have to be "sum of unknown coefficient times feature vector". Rewrite it with different parameters (as we did in section 2.2.4 for a different model, and as you had to do for example sheet 1 question 7) i.e. in the form $\beta_0 + \beta_1 e_1$ for a feature vector e_1 that you should identify, use `sklearn` to compute $\hat{\beta}_0$ and $\hat{\beta}_1$, and translate back to get $\hat{\lambda}$ and $\hat{\mu}$.

Question 3. Fitting the rich model is much like question 2. Fitting the simpler model is a classic supervised regression task, of the sort we've done many times already. Be careful about the vector sizes! For the rich model, we can only use responses $\texttt{Temp}_2, \ldots, \texttt{Temp}_n$. For the simpler model, we can use the all n records.

For the hypothesis test: call the rich model H_1, and think of the simpler model (call it H_0) as a restriction on the parameters of H_1, namely the restriction that $\lambda = 0$. What test statistic do you think would be useful here? Look back at example sheet 3 question 5.

Question 4. There's a brute force solution, very similar to example 11.2.1 from lecture notes: first use the law of total probability to condition on X_1 AND X_2, giving us an expression that includes $\mathbb{P}(X_2 = x_2, X_1 = x_1 \mid X_0 = g)$, and then break this expression down further using the definition of conditional probability with baggage $\{X_0 = g\}$.

There's also a more elegant solution based on a more sophisticated use of memorylessness, which says in its most general form that "conditional on the present, the past and the future are independent". This includes the sort of equation stated at the top of section 11.2 of lecture notes,

$$\mathbb{P}(X_3 = x_3 \mid X_2 = x_2, X_1 = x_1, X_0 = x_0) = \mathbb{P}(X_3 = x_3 \mid X_2 = x_2),$$

but it also includes situations where there are gaps in the future e.g.

$$\mathbb{P}(X_3 = x_3 \mid X_1 = x_1, X_0 = x_0) = \mathbb{P}(X_3 = x_3 \mid X_1 = x_1) \quad \text{(present is } x_1\text{)}$$

and situations where there are gaps in the past, e.g.

$$\mathbb{P}(X_3 = x_3 \mid X_2 = x_2, X_0 = x_0) = \mathbb{P}(X_3 = x_3 \mid X_2 = x_2) \quad \text{(present is } x_2\text{)}.$$

Can you use the gaps-in-the-past version to answer this question? Can you prove the gaps-in-the-past and the gaps-in-the-future versions of memorylessness?

Question 5. This is very similar to an example from lecture 15, in which we calculated

$$\mathbb{P}\big(X_1 = \text{drizzle} \mid X_0 = \text{rain}, X_2 = \text{rain}\big).$$

Just use likelihood notation (Pr) instead of probability ($\mathbb{P}$), to accommodate the fact that X and Z are continuous random variables. (All the standard laws of probability still hold when we're working with Pr; see lecture notes section 11.2 for a little more discussion.)

The last part of the question is asking you to take limits as $\varepsilon \to 0$ and as $\varepsilon \to \infty$. As a sanity check, try to give an intuitive explanation of your answer, in terms of how you'd predict Y in the case where (a) Z is almost noiseless and (b) Z is very noisy.

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, $P_{xy} = 1_{x \leftrightarrow y}/n_x$.

In general it's a good idea to try to solve the detailed balance equations first, and only if that fails is it worth trying to solve the full stationarity equations. In this case the detailed balance equations do indeed work.

Can you find the solution for a general connected undirected graph?

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: *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. This is a hidden Markov model, as mentioned in section 12.3:

$$\begin{array}{ccccccc} X_0 & \longrightarrow & X_1 & \longrightarrow & X_2 & \longrightarrow & \cdots \\ \downarrow & & \downarrow & & \downarrow & & \\ Y_0 & & Y_1 & & Y_2 & & \end{array}$$

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 11.2. 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.