Example sheet 4

Random systems

Data Science—DJW—2026/2027

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 (Fitting a Markov chain). 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, 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 2*. 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_climate0`****,* ***`fit_climate1`****, and* ***`test_climate0`****.]*

Question 3. We're given a dataset of strings $\boldsymbol{x}^{(i)}$, $i = 1, \ldots, n$ from alphabet $\mathcal{W}$, each string consisting of a sequence of tokens, $\boldsymbol{x}^{(i)} = x_1^{(i)} x_2^{(i)} \cdots x_{\ell_i}^{(i)}$ where ℓ_i is the length of string i and $x_j^{(i)} \in \mathcal{W}$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, \ell_i$. We decide to model them using a Markov chain on an augmented alphabet $\mathcal{W} \cup \{\square\}$ where $\square$ denotes 'end string': start at $\square$, then generate tokens until we hit $\square$, and take our generated string $\boldsymbol{X}$ to be the tokens between the two $\square$s.

Using the probability model $\mathbb{P}(X_{t+1} = v \,|\, X_t = u) = \theta_{u,v}$ where θ is a $(|\mathcal{W}|+1) \times (|\mathcal{W}|+1)$ matrix, calculate the maximum likelihood estimator for θ.

Question 4 (Causal diagrams). 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?

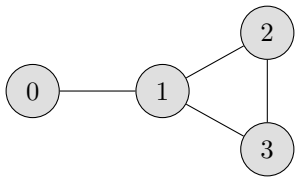
Question 5 (Markov chains). Draw the state space diagram for this Markov chain.

```
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 6. For the Cambridge weather simulator, as seen in lectures, show that

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

Question 7 (Stationarity). 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 8*. 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 9 (Hidden Markov model). 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 $\Pr(x_{n-1} \mid h)$.

(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 10. The code from question 9 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 9(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. Follow the same pattern as in lectures. 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", whereas this model is $(\mu - \lambda\mu)1 + \lambda x_{i-1}$. Choose a different parameterization that makes this a proper linear model (as we did in section 2.2.4 for a periodic model, and as you had to do for example sheet 1 question 8 for a piecewise linear model) 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 using the plug-in principle (as in example sheet 1 question 5) to get $\hat{\lambda}$ and $\hat{\mu}$.

Question 2. Fitting the rich model is much like question 1. 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 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?

Question 3. The answer is intuitively obvious, and you should remember it from MLRD tick 7:

$$\hat{\theta}_{uv} = \frac{\#\text{ transitions } u \to v}{\#\text{ transitions from } u}.$$

In MLRD, you just wrote down the answer, because it's obvious! This question asks you to obtain it formally, as a result of likelihood maximization. Take care to include $\square$ properly in the likelihood expression, and think about the constraints on the parameters you wish to estimate. Exercise 1.3.5 from lecture notes, on fitting a categorical sample, is relevant.

Question 4. This is very similar to an example from lecture 15, in which we calculated

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

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.)

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 5. 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 6. There's a brute force solution, very similar to the solution from lectures: first use the law of total probability to condition on X_1 AND X_2, giving us an expression that includes $\mathbb{P}(X_2 = x_2, X_1 = x_1 | X_0 = g)$, 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 simple statements about the immediate future,

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

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 7. 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 8. 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 calculation in lectures 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 9. This is a hidden Markov model, familiar to you from IA MLRD:

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

Part (a). The second equation requires the law of total probability (with baggage), and the third equation requires Bayes's rule (with baggage h). 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 supplementary questions are not intended for supervision (unless your supervisor directs you otherwise).

Question 11 (Reinforcement learning). A drone is trying to hover at a target height h_0, but random gusts of wind push it up and down. Let X_t be its height at time t, starting at $X_0 = h_0$, and suppose it uses a control policy that applies a thrust proportional to its distance from the target:

$$X_{t+1} = X_t - \theta(X_t - h_0) + N(0, \sigma^2)$$

where the noise term represents random gusts and σ is known. Here $\theta \in \mathbb{R}_{>0}$ is the control parameter, specifying how aggressively the drone responds; we wish to tune θ to optimize performance. Performance is measured as a cost: the cost $c_\theta(\boldsymbol{x})$ of a sequence of states $\boldsymbol{x} = x_0, \dots, x_T$ over a fixed horizon T comprises a fuel cost plus a penalty for deviating from the target,

$$c_\theta(\boldsymbol{x}) = \sum_{t=0}^{T-1} \left(w_{\text{fuel}}\, \theta\, |x_t - h_0| + w_{\text{dev}}\, (x_{t+1} - h_0)^2 \right)$$

where w_{fuel} and w_{dev} are given weight parameters. The expected cost is $\mathbb{E}\, c_\theta(\boldsymbol{X})$, which is a function of θ both directly in the cost function and also because the distribution of $\boldsymbol{X}$ depends on θ.

(a) Write an expression for $\log \Pr(\boldsymbol{x}; \theta)$.

(b) By writing $\mathbb{E}\, c_\theta(\boldsymbol{X})$ as an integral over trajectories $\boldsymbol{x} \in \mathbb{R}^T$, show that

$$\frac{d}{d\theta}\, \mathbb{E}\, c_\theta(\boldsymbol{X}) = \mathbb{E}\Big[\frac{\partial}{\partial \theta} c_\theta(\boldsymbol{X}) + c_\theta(\boldsymbol{X}) \frac{\partial}{\partial \theta} \log \Pr(\boldsymbol{X}; \theta) \Big].$$

(c) Explain how to estimate this gradient from trajectories $\boldsymbol{x}$ sampled at the current θ, and hence how to improve θ.

Hint for Question 11. *For the second term in part (b), you may be able to guess the algebraic step from the expression you're asked to derive; if not, look up the "log derivative trick".*

This question is a simple example of the policy gradient *method in reinforcement learning. It appears as the alignment step in training large language models, under the heading RLHF (reinforcement learning from human feedback). Some further reading:*

- *An MPhil course, L170 "Advanced Reinforcement Learning".*
- *A blog post by Andrej Karpathy, "Deep reinforcement learning: Pong from pixels"*
- *A sceptical view from a control theorist, Ben Recht: "An outsider's tour of reinforcement learning part 6".*

In these accounts the controller picks its actions at random, and the log likelihood being differentiated is that of its random choices. In this question the drone's response is deterministic, and the randomness is all in the gusts.

Question 12 (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

$$\pi_v = \frac{1-d}{|V|} + d \sum_{u : u \to v} \frac{\pi_u}{|\Gamma_u|} + \frac{d}{|V|} \sum_{u : |\Gamma_u| = 0} \pi_u$$

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 a simple web graph of your choice. 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 13. The Markov model for motion from question 5 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.

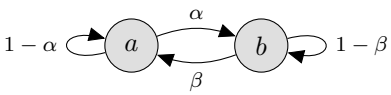
(a) Let $V_n \in \{-1, 0, 1\}$ be a Markov chain: let $V_{n+1} = V_n$ with probability 0.9, and let V_{n+1} be chosen uniformly at random from $\{-1, 0, 1\}$ with probability 0.1. Draw a state space diagram for this Markov chain.

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.

Question 14. Here is the state space diagram for a Markov chain. Find a stationary distribution. Is it unique?



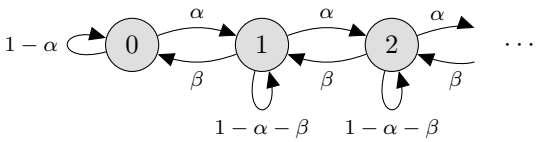
Hint for Question 14. *The equations to solve are*

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

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

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

Question 15. 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.



Hint for Question 15. *When a question asks "show that π is a stationary distribution", you don't have to set about trying to* solve *the equations*

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

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 ($\sum_{n=0}^{\infty} ar^n = a/(1-r)$ 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.)