Practical 4

Foundations of Data Science—DJW—2019/2020

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

Practical 4 can be found on Azure Notebooks, `prac4.ipynb`. In it, you will implement a particle filter for estimating the location of a moving object, given noisy readings. The following questions illustrate how the computation works, but in a simpler setting where it's possible to write out exact formulae. I recommend you answer these questions first, before embarking on the practical.

Consider the following code. It computes a sequence of x values, exactly the same as in example sheet 4 question 3, and this sequence is a Markov chain. But we don't observe x directly, we only see noisy observations $x + e$. This is called a *hidden Markov model*.

```
def hmm():
    MAX_STATE = 9
    # Pick the initial state uniformly from
    x = numpy.random.randint(low=0, high=MAX_STATE+1)
    while True:
        e = numpy.random.choice([−1,0,1])
        yield min(MAX_STATE, max(0, x + e))
        d = numpy.random.choice([−1,0,1], p=[1/4,1/2,1/4])
        x = min(MAX_STATE, max(0, x + d))
```

Question 1. Let $X = (X_0, X_1, \dots)$ be the sequence of x values computed inside this code, and let $Y = (Y_0, Y_1, \dots)$ be the observations, where Y_n is X_n plus noise. Draw the state space diagram for X. Draw a causal diagram for $\{X_0, Y_0, X_1, Y_1, X_2, \dots\}$.

Question 2. Define $\pi_x^{(0)}$ and $\delta_x^{(0)}$ by

$$\begin{aligned} \delta_x^{(0)} &= \mathbb{P}(X_0 = x) = 1/10 \quad \text{for } x \in \{0, \dots, 9\} \\ \pi_x^{(0)} &= \mathbb{P}(X_0 = x \mid Y_0 = y). \end{aligned}$$

(Since $\pi_x^{(0)}$ depends on y_0 we ought to write it as a function of x and y_0, but for the sake of conciseness we won't write out the y_0 dependency.) Use Bayes's rule to find a formula for $\pi_x^{(0)}$ in terms of $\delta^{(0)}$ and the matrix $Q_{xy} = \mathbb{P}(Y_n = y \mid X_n = x)$.

Question 3. Let $\delta_x^{(1)} = \mathbb{P}(X_1 = x \mid Y_0 = y_0)$. Use the law of total probability to find a formula for $\delta_x^{(1)}$ in terms of $\pi^{(0)}$ and the transition matrix $P_{xx'} = \mathbb{P}(X_{n+1} = x' \mid X_n = x)$.

Question 4. Let

$$\begin{aligned} \delta_x^{(n)} &= \mathbb{P}(X_n = x \mid Y_0 = y_0, \dots, Y_{n-1} = y_{n-1}) \\ \pi_x^{(n)} &= \mathbb{P}(X_n = x \mid Y_0 = y_0, \dots, Y_{n-1} = y_{n-1}, Y_n = y_n) \end{aligned}$$

(a) Show that

$$\pi_x^{(n)} = \frac{\delta_x^{(n)} Q_{xy}}{\sum_z \delta_z^{(n)} Q_{zy}} \qquad \text{and} \qquad \delta_x^{(n)} = \sum_z \pi_z^{(n-1)} P_{zx}.$$

(b) Write pseudocode for a function that takes as input a list of readings $y = [y_0, y_1, \dots, y_n]$ and outputs the vector $\pi^{(n)}$. Your pseudocode should include defining the P and Q matrices.

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

Solutions

Question 1. From lecture notes section 9.1,

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

Question 2. Using Bayes's rule,

$$\pi_x^{(0)} = \mathbb{P}(X_0 = x \mid Y_0 = y) = \text{const} \times \mathbb{P}(X_0 = x)\,\mathbb{P}(Y_0 = y \mid X_0 = x) = \text{const} \times \delta_x^{(0)} Q_{xy_0}$$

where the constant is whatever makes $\sum_x \pi_x^{(0)} = 1$, thus

$$\pi_x^{(0)} = \frac{\delta_x^{(0)} Q_{xy_0}}{\sum_{x'} \delta_{x'}^{(0)} Q_{x'y_0}}$$

Question 3.

$$\begin{aligned}
\delta_x^{(1)} &= \mathbb{P}(X_1 = x \mid Y_0 = y_0) \\
&= \sum_{x_0} \mathbb{P}(X_1 = x \mid X_0 = x_0, Y_0 = y_0)\,\mathbb{P}(X_0 = x_0 \mid Y_0 = y_0) \quad \text{by law tot. prob.} \\
&= \sum_{x_0} \mathbb{P}(X_1 = x \mid X_0 = x_0)\,\mathbb{P}(X_0 = x_0 \mid Y_0 = y_0) \quad \text{from causal diagram} \\
&= \sum_{x_0} \pi_{x_0}^{(0)} P_{x_0 x}.
\end{aligned}$$

Question 4. The two equations come from the same reasoning as above. For the code,

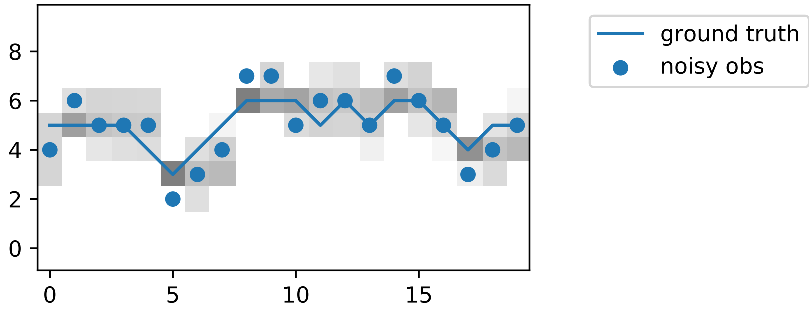
```
# Emission matrix.
# Q[x,y] = Prob(reading is y | true position is x)
Q = np.zeros((10,10))
for x in range(10):
    Q[x, max(x-1,0)] += 1/3
    Q[x, x] += 1/3
    Q[x, min(x+1,9)] += 1/3
assert all(np.sum(Q, axis=1) == 1)

# Transition matrix
# P[x,y] = Prob(next position is y | current position is x)
P = np.zeros((10,10))
for x in range(10):
    P[x, max(x-1,0)] += 1/4
    P[x,x] += 1/2
    P[x, min(x+1,9)] += 1/4
assert all(np.sum(P, axis=1) == 1)

δ0 = np.ones(10) / 10    # initial distribution, uniform on {0,1,...,9}

def filter_obs(δ0, ys, P, Q):
    δ = δ0
    for y in ys:
        π = Q[:, y] * δ
        π = π / sum(π)
        δ = π @ P
    return π
```

Here is the result of a simulation. The plot shows the simulated values of X_n labelled 'ground truth', the simulated observations Y_n labelled 'noisy obs', and the $\pi^{(n)}$ vector at each timestep, indicated by shading.



When the code is run on inputs $[3, 4, 4, 9]$ it fails because the denominator is 0 when normalizing $\pi^{(4)}$. Concretely, it's impossible to see observation $Y_3 = 4$ (which implies $X_3 \in \{3, 4, 5\}$) followed by $Y_4 = 9$ (which implies $X_4 \in \{8, 9\}$). So, when we use Bayes's rule to derive $\pi^{(4)}$, we're conditioning on an event with probability 0, and so Bayes's rule doesn't apply.