UNIVERSITY OF PENNSYLVANIA

ESE 650: LEARNING IN ROBOTICS

SPRING 2021

[01/29] HOMEWORK 1

DUE: 02/12 FRI 11.59 PM

Changelog: This space will be used to note down updates/errata to the homework problems.

Instructions

Read the following instructions carefully before beginning to work on the homework.

- You will submit solutions typeset in LaTeX on Gradescope (strongly encouraged). You can use hw_template.tex on Canvas in the "Homeworks" folder to do so. If your handwriting is *unambiguously legible*, you can submit PDF scans/tablet-created PDFs.
- Please start a new problem on a fresh page and mark all the pages corresponding to each problem. Failure to do so may result in your work not graded completely.
- Clearly indicate the name and Penn email ID of all your collaborators on your submitted solutions.
- **For each problem in the homework, you should mention the total amount of time you spent on it. This helps us gauge the perceived difficulty of the problems.**
- You can be informal while typesetting the solutions, e.g., if you want to draw a picture feel free to draw it on paper clearly, click a picture and include it in your solution. Do not spend undue time on typesetting solutions.
- You will see an entry of the form "HW 1 PDF" where you will upload the PDF of your solutions. You will also see entries like "HW 1 Problem 1 Code" where you will upload your solution for the respective problems. **For each programming problem, you should create a fresh Python file**. This file should contain all the code to reproduce the results of the problem and you will upload the .py file to Gradescope. If we have installed Autograder for a particular problem, you will use the Autograder. Name your file to be "pennkey_hw1_problem1.py", e.g., I will name my code for Problem 1 as "pratikac_hw1_problem1.py".

- **You should include all the relevant plots in the PDF, without doing so you will not get full credit.** You can, for instance, export your Jupyter notebook as a PDF (you can also use text cells to write your solutions) and export the same notebook as a Python file to upload your code.
- **Your PDF solutions should be completely self-contained. We will run the Python file to check if your solution reproduces the results in the PDF.**

Credit The points for the problems add up to 120. You only need to solve for 100 points to get full credit, i.e., your final score will be $\min(\text{your total points}, 100)$.

Problem 1 (50 points, Bayes filter). In this problem we are given a robot operating in a 2D grid world. Every cell in the grid world is characterized by a color (0 or 1). The robot is equipped with a noisy odometer and a noisy color sensor. Given a stream of actions and corresponding observations, implement a Bayes filter to keep track of the robot's current position. The sensor reads the color of cell of the grid world correctly with probability 0.9 and incorrectly with probability 0.1. At each step, the robot can take an action to move in 4 directions (north, east, south, west). Execution of these actions is noisy, so after the robot performs this action, it actually makes the move with probability 0.9 and stays at the same spot without moving with probability 0.1.

When the robot is at the edge of the grid world and is tasked with executing an action that would take it outside the boundaries of the grid world, the robot remains in the same state with probability 1. Start with a uniform prior on all states. For example if you have a world with 4 states (x_1, x_2, x_3, x_4) then $\mathrm{P}(X_0 = x_1) = \mathrm{P}(X_0 = x_2) = \mathrm{P}(X_0 = x_3) = \mathrm{P}(X_0 = x_4) = 0.25$.

You are given a zip file with some starter code for this problem: this consists of the Python scripts example_test.py and histogram_filter.py (Bayes filter is also called the Histogram filter) and a starter file containing some data: starter.npz. The starter.npz file contains a binary color-map (the grid), a sequence of actions, a sequence of observations, and a sequence of the correct belief states. This is provided for you to debug your code. You should implement your code in the histogram_filter.py. Be careful not to change the function signature, or your code will not pass the tests on the autograder. You will turn in your assignment using Gradescope.

Problem 2 (35 points, Learning HMMs). In this problem, we are going to implement what is called the Baum-Welch algorithm for HMMs. Recall that an HMM with observation matrix M has an underlying Markov chain with an initial distribution π and state-transition matrix T. Let us denote our HMM by

$$\lambda = (\pi, T, M).$$

Say someone had given us an HMM λ which gave us a sequence of observations $Y_1, \ldots, Y_t$. Since these observations happened, they tell us something more about our given state-transition and observation matrices, for example, given these observations, we can go back and modify T, M to be such that the observation sequence is more likely, i.e., it improves

$$\mathrm{P}(Y_1, \ldots, Y_t \mid \lambda).$$

(a) **(20 points)** Assume that we are tracking a criminal who shuttles merchandise between Los Angeles (x_1) and New York (x_2). The state transition matrix between these states is

$$T = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix};$$

e.g., given the person is in LA, he is likely to stay in LA or go to NY with equal probability. We can make observations about this person, we either observe him to be in LA (y_1), NY (y_2) or do not observe anything at all (null, y_3).

$$M = \begin{bmatrix} 0.4 & 0.1 & 0.5 \\ 0.1 & 0.5 & 0.4 \end{bmatrix}.$$

Say that we are tracking the person and we observed an observation sequence of 20 periods

$$(\text{null, LA, LA, null, NY, null, NY, NY, NY, null, NY, NY, NY, NY, NY, null, null, LA, LA, NY}).$$

Assume that we do not know where the criminal is at the first time-step, so the initial distribution π is uniform.

We first compute the probability of each state given all the observations, i.e.,

$$\gamma_k(x) = \mathrm{P}(X_k = x \mid Y_1, \dots, Y_t, \lambda) = \frac{\alpha_k(x)\ \beta_k(x)}{\sum_x \alpha_t(x)}.$$

which is simply our smoothing probability computed using the HMM λ computed using the forward and backward variables α_k, β_k. Compute the smoothing probabilities $\gamma_k(x)$ for all 20 time-steps and both states and show it in a table like we had in the lecture notes. You should get the point-wise most likely sequence of states after doing so to be

$$(\text{LA, LA, LA, LA, NY, LA, NY, NY, NY, LA, NY, NY, NY, NY, NY, NY, LA, LA, LA, NY}).$$

Notice how smoothing fills in the missing observations above.

(b) **(5 points)** We now discuss how to update the model λ given our observations. First observe that

$$\mathrm{E}[\text{number of times the Markov chain was in state } x] = \sum_{k=1}^{t-1} \gamma_k(x).$$

Let us define a new quantity

$$\xi_k(x, x') = \mathrm{P}(X_k = x, X_{k+1} = x' \mid Y_1, \dots, Y_t, \lambda)$$

to be the probability of being at a state x at time k and then moving to state x' at time $k+1$, conditional upon all the observations we have received from our HMM. Show that

$$\xi_k(x, x') = \eta\ \alpha_k(x) T_{x,x'} M_{x',y_{k+1}}\ \beta_{k+1}(x')$$

where η is a normalizing constant that makes $\sum_{x,x'} \xi_k(x, x') = 1$.

(c) **(5 points)** We can now use our estimate from the previous part in to get

$$\mathrm{E}[\text{number of transitions from } x \text{ to } x'] = \sum_{k=1}^{t-1} \xi_k(x, x').$$

We now construct an updated model for our HMM as follows. The initial distribution of the states, instead of being π is our smoothing probability

$$\pi' = \gamma_1(x).$$

Entries of the new transition matrix can be computed by

$$T'_{x,x'} = \frac{\text{E[number of transitions from } x \text{ to } x']}{\text{E[number of times the Markov chain was in state } x]} = \frac{\sum_{k=1}^{t-1} \xi_k(x, x')}{\sum_{k=1}^{t-1} \gamma_k(x)}.$$

Entries of the new observation matrix can be computed by

$$M'_{x,y} = \frac{\text{E[number of times in state } x, \text{ when observation was } y]}{\text{E[number of times the Markov chain was in state } x]} = \frac{\sum_{k=1}^{t} \gamma_k(x) \mathbf{1}_{\{y_k = y\}}}{\sum_{k=1}^{t-1} \gamma_k(x)}.$$

where $\mathbf{1}_{\{y_k=y\}}$ denotes that the observation at the k^{th} timestep was y. Let the new HMM model be

$$\lambda' = (\pi', T', M')$$

Write down the new HMM model λ' and see if any entires have changed as compared to λ.

(d) **(5 points)** Compare the two HMMs λ and λ'. Compute the two probabilities and show that

$$\mathrm{P}(Y_1, \ldots, Y_t \mid \lambda) < \mathrm{P}(Y_1, \ldots, Y_t \mid \lambda').$$

This is exactly what Baum et al. proved in the paper Baum, Leonard E., et al. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." The annals of mathematical statistics 41.1 (1970): 164-171. You can see that the Baum-Welch algorithm is quite powerful to learn real-world Markov chains, e.g., the gait of your dog. Given some elementary model of the dynamics T and an observation model M, you can sequentially update both of them to guess a better model of the dynamics.

Problem 3 (20 points, Forward-Backward algorithm). Answer the following questions.

(a) **(15 points)** Using our forward and backward variables

$$\alpha_k(x) = \mathrm{P}(Y_1, \ldots, Y_k, X_k = x)$$
$$\beta_k(x) = \mathrm{P}(Y_{k+1}, \ldots, Y_t \mid X_k = x).$$

for a sequence of observations $Y_1, \ldots, Y_t$ of length $t > 1$, the state transition matrix $T_{ij} = \mathrm{P}(X_{k+1} = x_j \mid X_k = x_i)$ and the observation matrix $M_{ij} = \mathrm{P}(Y_k = y_j \mid X_k = x_i)$, write down how you will compute the following probabilities

(1) $\mathrm{P}(X_{k+1} = x_j \mid X_k = x_i,\ Y_1, \ldots, Y_t)$,
(2) $\mathrm{P}(X_k = x_i \mid X_{k+1} = x_j,\ Y_1, \ldots, Y_t)$,
(3) $\mathrm{P}(X_{k-1} = x_i, X_k = x_j, X_{k+1} = x_l \mid Y_1, \ldots, Y_t)$.

You do not need to consider the boundary cases like $k \in \{1, t\}$.

(b) **(5 points)** Viterbi's algorithm finds the most likely state trajectory of the HMM's associated Markov chain given a sequence of observations $Y_1, \ldots, Y_t$. Explain why, in general, the solution of the decoding problem, i.e.,

$$(x_1^*, \ldots, x_t^*) = \underset{(x_1, \ldots, x_t)}{\operatorname{argmax}}\ \mathrm{P}(X_1 = x_1, \ldots, X_t = x_t \mid Y_1, \ldots, Y_t).$$

is not the same as the sequence obtained by most likely state at each time computed by the smoothing, i.e.,

$$(\hat{x_1}, \ldots, \hat{x_t}) \text{ where } \hat{x_k} = \operatorname*{argmax}_x \mathrm{P}(X_k = x \mid Y_1, \ldots, Y_t).$$

Give an example where the two *are the same*.

Problem 4 (15 points, Optimal estimation). We will study the simplest case of a filtering problem, namely estimation of a static, scalar variable $X \in \mathbb{R}$. We take two noisy measurements of the scalar X of the form

$$Y_i = h_i X + \epsilon_i; \quad i = 1, 2$$

where $h_1 = 1$ and $h_2 = 2$. The noise in our measurements $\epsilon_i \in \mathbb{R}$ are distributed as

$$\mathrm{E}[\epsilon_i] = 0, \text{ and } \mathrm{E}[\epsilon_i^2] = \sigma_i^2.$$

The two noise terms are uncorrelated with each other. Assume that our signal X is not correlated with noise $\mathrm{E}[X\epsilon_i] = 0$ for both $i = 1, 2$.

(a) **(10 points)** Assume that the optimal estimate of the variable is of the form

$$\hat{X} = a_1 Y_1 + a_2 Y_2$$

where constants a_1 and a_2 are independent of X. Compute the values of a_1, a_2 that

(i) make sure that the estimate $\hat{X}$ is unbiased, and

(ii) minimize the mean-square estimation error $\mathrm{E}[(X - \hat{X})^2]$.

Use these values of a_1, a_2 to compute the minimum value of the mean-square estimation error. You should solve this problem from first principles, without using any expressions for the Kalman filter that we may derive in the class.

(b) **(5 points)** Discuss how the optimal estimator uses the two measurements for the following cases

(i) $\sigma_2 \gg \sigma_1$

(ii) $\sigma_2 = \sigma_1$

(iii) $\sigma_2 \ll \sigma_1$.

Do your answers agree with your intuition? Explain.