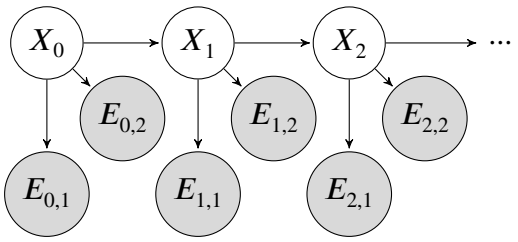
CS 188 Spring 2023 Introduction to Artificial Intelligence

Exam Prep 10 Solutions

Q1. Particle Filtering

You've chased your arch-nemesis Leland to the Stanford quad. You enlist two robo-watchmen to help find him! The grid below shows the campus, with ID numbers to label each region. Leland will be moving around the campus. His location at time step t will be represented by random variable X_t. Your robo-watchmen will also be on campus, but their locations will be fixed. Robot 1 is always in region 1 and robot 2 is always in region 9. (See the * locations on the map.) At each time step, each robot gives you a sensor reading to help you determine where Leland is. The sensor reading of robot 1 at time step t is represented by the random variable $E_{t,1}$. Similarly, robot 2's sensor reading at time step t is $E_{t,2}$. The Bayes Net to the right shows your model of Leland's location and your robots' sensor readings.

1*	2	3	4	5
6	7	8	9*	10
11	12	13	14	15



In each time step, Leland will either stay in the same region or move to an adjacent region. For example, the available actions from region 4 are (WEST, EAST, SOUTH, STAY). He chooses between all available actions with equal probability, regardless of where your robots are. Note: moving off the grid is not considered an available action.

Each robot will detect if Leland is in an adjacent region. For example, the regions adjacent to region 1 are 1, 2, and 6. If Leland is in an adjacent region, then the robot will report $NEAR$ with probability 0.8. If Leland is not in an adjacent region, then the robot will still report $NEAR$, but with probability 0.3.

For example, if Leland is in region 1 at time step t the probability tables are:

E	$P(E_{t,1}\|X_t = 1)$	$P(E_{t,2}\|X_t = 1)$
$NEAR$	0.8	0.3
FAR	0.2	0.7

(a) Suppose we are running particle filtering to track Leland's location, and we start at $t = 0$ with particles [$X = 6$, $X = 14$, $X = 9$, $X = 6$]. Apply a forward simulation update to each of the particles using the random numbers in the table below.

Assign region IDs to sample spaces in numerical order. For example, if, for a particular particle, there were three possible successor regions 10, 14 and 15, with associated probabilities, $P(X = 10)$, $P(X = 14)$ and $P(X = 15)$, and the random number was 0.6, then 10 should be selected if $0.6 \leq P(X = 10)$, 14 should be selected if $P(X = 10) < 0.6 < P(X = 10) + P(X = 14)$, and 15 should be selected otherwise.

Particle at $t = 0$	Random number for update	Particle after forward simulation update
$X = 6$	0.864	11
$X = 14$	0.178	9
$X = 9$	0.956	14
$X = 6$	0.790	11

(b) Some time passes and you now have particles $[X = 6, X = 1, X = 7, X = 8]$ at the particular time step, but you have not yet incorporated your sensor readings at that time step. Your robots are still in regions 1 and 9, and both report $NEAR$. What weight do we assign to each particle in order to incorporate this evidence?

Particle	Weight
$X = 6$	0.8 * 0.3
$X = 1$	0.8 * 0.3
$X = 7$	0.3 * 0.3
$X = 8$	0.3 * 0.8

(c) To decouple this question from the previous question, let's say you just incorporated the sensor readings and found the following weights for each particle (these are not the correct answers to the previous problem!):

Particle	Weight
$X = 6$	0.1
$X = 1$	0.4
$X = 7$	0.1
$X = 8$	0.2

Normalizing gives us the distribution

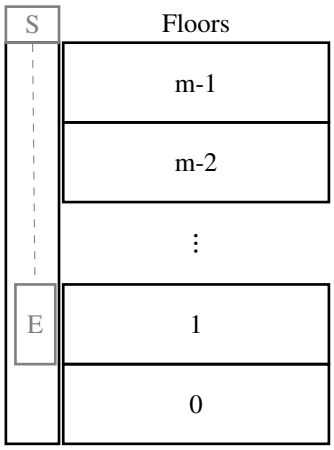
$$X = 1 : 0.4/0.8 = 0.5$$
$$X = 6 : 0.1/0.8 = 0.125$$
$$X = 7 : 0.1/0.8 = 0.125$$
$$X = 8 : 0.2/0.8 = 0.25$$

Use the following random numbers to resample you particles. As on the previous page, **assign region IDs to sample spaces in numerical order.**

Random number:	0.596	0.289	0.058	0.765
Particle:	6	1	1	8

Q2. Decode Your Terror (HMM)

You go to Disney and ride the famous Tower of Terror ride, where an elevator rises and drops seemingly at random. You're terrified, but vow to determine the sequence of rises and drops that make up the ride so you won't be as terrified next time. Assume the elevator E follows a Markovian process and it has m floors at which it can stop. In the dead of night, you install a sensor S at the top of the shaft that gives approximate distance measurements, quantized into n different distance bins. Assume that the elevator stops at T floors as part of the ride and the initial distribution of the elevator is uniform over the m floors.



You want to know the most probable sequence of (hidden) states $X_{1:T}$ given your observations $y_{1:T}$ from the sensor, so you turn to the Viterbi algorithm, which performs the following update at each step:

$$m_t[x_t] = P(y_t|x_t) \max_{x_{t-1}} \left[P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]\right]$$
$$a_t[x_t] = \arg\max_{x_{t-1}} m_{t-1}[x_{t-1}]$$

(a) **(i)** What is the run time of the Viterbi algorithm to determine all previous states for this scenario? Please answer in big O notation, in terms of T, m, and n, or write "N/A" if the run time is unable to be determined with the given information.

Tm^2

For each timestep that we want to decode, we loop over all states and consider the transition probability to that state from all previous states. Thus, the runtime is Tm^2, since we repeat this process for each timestep (T), and at each timestep we have a double for loop for states (one to compute m-value for all states, one to compute arg max of previous m-values and transition probabilities, so m^2). For more details, see Note 6 (pg. 8).

(ii) What is the space complexity of the Viterbi algorithm to determine all previous states for this scenario? Please answer in big O notation, in terms of T, m, and n, or write "N/A" if the space complexity is unable to be determined with the given information.

Tm

We store only our a-value arrays (which keep track of the best transition along the path to the current state). These are essentially back-pointers so we can reconstruct the best path. For a sequence of length T, we then have a matrix of these pointers that is $T \times m$, so we take up Tm space.

Eventually, we decide that the end of the ride is the exciting part, so we decide that we only wish to determine the previous K states.

(b) **(i)** What is the run time of the Viterbi algorithm to determine the previous K states? Please answer in big O notation, in terms of T, K, m, and n, or write "N/A" if the run time is unable to be determined with the given information.

Tm^2

The run time is the same as (a)(i) since the forward pass part of the Viterbi algorithm is unchanged (e.g., we still need to consider all transition probabilities at all timesteps to determine the most likely path, regardless of how far back we want to go).

(ii) What is the space complexity of the Viterbi algorithm to determine the previous K states? Please answer in big O notation, in terms of T, K, m, and n, or write "N/A" if the space complexity is unable to be determined with the given information.

Km

If we only wish to determine the previous K states, we only need to store a-values corresponding to the previous K timesteps (since when we reconstruct the path, we only need to go K steps back). Thus, the space complexity decreases to Km from part (a)(ii).

Suppose you instead only wish to determine the current distribution (at time T) for the elevator, given your T observations, so you use the forward algorithm, with update step shown here:

$$P(X_t|y_t) \propto P(y_t|x_t) \sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}, y_{0:t-1})$$

Additionally, from your previous analysis, you note that there are some states which are unreachable from others (e.g., the elevator cannot travel from the top floor to the bottom in a single timestep). Specifically, from each state, there are between $G/2$ and G states which can be reached in the next timestep, where $G < m$.

(c) **(i)** What is the run time for the forward algorithm to estimate the current state at time T, assuming we ignore states that cannot be reached in each update? Please answer in big O notation, in terms of T, m, G, and n, or write "N/A" if the run time is unable to be determined with the given information.

TmG

The normal run time for the forward algorithm is Tm^2 (very similar to Viterbi - we consider transition probabilities from each state to each other state during our forward pass). However, if we can ignore those states with zero transition probability, then we only need to consider G states when calculating our sum for each state. Thus, the run time is reduced to TmG.

(ii) What is the space complexity for the forward algorithm to estimate the current state at time T, assuming we ignore states that cannot be reached in each update? Please answer in big O notation, in terms of T, m, G, and n, or write "N/A" if the space complexity is unable to be determined with the given information.

m

The space complexity for the forward algorithm is unchanged by how many states we can reach from each state, so this is the same as the normal space complexity for the forward algorithm. It is m because we store $P(X_t|y_t)$ for

each state and we only store one timestep (since we only care about the current state probabilities).

Finally, assume that the number of elevator states is actually infinite (e.g., instead of stopping at floors, the elevator can stop at any point along the elevator shaft).

(d) What is the run time of the standard forward algorithm to estimate the current state at time T in this case? Please answer in big O notation, in terms of T, m, G, and n, or write "N/A" if the run time is unable to be determined with the given information.

N/A

From part (c)(i), the run time of the forward algorithm is proportional to the number of states m. Thus as $m \to \infty$, the run time grows to infinity. Both N/A and ∞ were accepted here.

(e) Suppose you decide to use a particle filter instead of the forward algorithm. What is the run time of a particle filter with P particles? Please answer in big O notation, in terms of T, m, G, n, and P, or write "N/A" if the run time is unable to be determined with the given information.

TP

Assuming the time for resampling a single particle is constant, then the runtime is TP. At each timestep (total of T), we perform the time elapse and observation updates in constant time for each of the P particles.