CS 486/686 Assignment 3 (85 marks)

Alice Gao

Due Date: 11:59 PM ET on Wednesday, November 24, 2021

Academic Integrity Statement

If your written submission on Learn does not include this academic integrity statement with your signature (typed name), we will deduct 5 marks from your final assignment mark.

I declare the following statements to be true:

- The work I submit here is entirely my own.
- I have not shared and will not share any of my code with anyone at any point.
- I have not posted and will not post my code on any public or private forum or website.
- I have not discussed and will not discuss the contents of this assessment with anyone at any point.
- I have not posted and will not post the contents of this assessment and its solutions on any public or private forum or website.
- I will not search for assessment solutions online.
- I am aware that misconduct related to assessments can result in significant penalties, possibly including failure in the course and suspension. This is covered in Policy 71: https://uwaterloo.ca/secretariat/policies-procedures-guidelines/policy-71.

Failure to accept the integrity policy will result in your assignment not being graded.

By typing or writing my full legal name below, I confirm that I have read and understood the academic integrity statement above.

Instructions

- Submit any written solutions in a file named `writeup.pdf` to the A3 Dropbox on `Learn`. Submit any code to `Marmoset` at https://marmoset.student.cs.uwaterloo.ca/. No late assignment will be accepted. This assignment is to be done individually.
- I strongly encourage you to complete your write-up in Latex, using this source file. If you do, in your submission, please replace the author with your name and student number. Please also remove the due date, the Instructions section, and the Learning goals section. Thanks!
- Lead TAs:
  - Ji Xin (ji.xin@uwaterloo.ca)
  - Blake Vanberlo (bvanberlo@uwaterloo.ca)

  The TAs' office hours will be posted on MS Teams.

Learning goals

Inference in Bayesian Networks

- Define factors. Manipulate factors using the operations restrict, sum out, multiply and normalize.
- Trace the execution of and implement the variable elimination algorithm for calculating a prior or a posterior probability given a Bayesian network.

Inference in Hidden Markov Models

- Construct a hidden Markov model given a real-world scenario.
- Perform filtering and smoothing by executing the forward-backward algorithm.

1 Locating the Robot via an HMM (14 marks)

You will calculate some probabilities for a simplified robot localization problem by executing the forward-backward algorithm.

The Simplified Robot Localization Problem:

The robot is in an environment with 2 cells labelled 0 and 1. Let S_t denote the robot's location at time step t.

The robot has a noisy sensor, which recognizes its current location with a probability of 0.8. Let O_t denote the robot's sensor reading at time step t. The observation probabilities are as follows.

$$P(O_t = 1|S_t = 1) = 0.8 \tag{1}$$
$$P(O_t = 0|S_t = 0) = 0.8 \tag{2}$$

The robot has one available action **Move**. If the robot takes the action, it has a 0.3 probability of staying in its current location and has a 0.7 probability of moving to the other location.

The effects of **Move** at time t:

$$P(S_{t+1} = 1|S_t = 0) = 0.7 \tag{3}$$
$$P(S_{t+1} = 0|S_t = 1) = 0.7 \tag{4}$$

We will represent this problem for three time steps (0 to 2) using the graphical model in Figure 1.

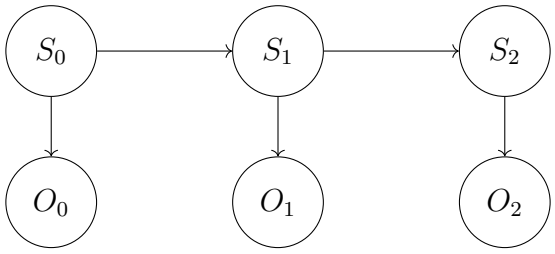

Figure 1: A Hidden Markov Model of Simplified Robot Localization Problem

Please answer the following questions.

(a) Answer the question below by executing the forward-backward algorithm. **Show all the steps. Make sure that your answer has 4 decimal places.**

Problem: At time step $k = 2$, what is the probability that the robot is in location 0?

You can use the following information in your calculations.

- At time step $k = 0$, the robot's prior belief over the two locations is a uniform distribution. That is, $P(S_0 = 0) = 0.5$ and $P(S_0 = 1) = 0.5$.
- At time step $k = 0$, the robot's sensor reading is $O_0 = 0$.
- At time step $k = 1$, the robot's sensor reading is $O_1 = 1$.
- At time step $k = 2$, the robot's sensor reading is $O_2 = 1$.

Marking Scheme: (8 marks) Correct formulas, calculations and final answer.

(b) Answer the question below by executing the forward-backward algorithm. **Show all the steps. Make sure that your answer has 4 decimal places.**

Problem: At time step $k = 1$, what is the probability that the robot is in location 0?

You can use the following information in your calculations.

- At time step $k = 0$, the robot's prior belief over the two locations is a uniform distribution. That is, $P(S_0 = 0) = 0.5$ and $P(S_0 = 1) = 0.5$.
- At time step $k = 0$, the robot's sensor reading is $O_t = 0$.
- At time step $k = 1$, the robot's sensor reading is $O_t = 1$.
- At time step $k = 2$, the robot's sensor reading is $O_t = 1$.

Marking Scheme: (6 marks) Correct formulas, calculations and final answer.

The Robot Environment

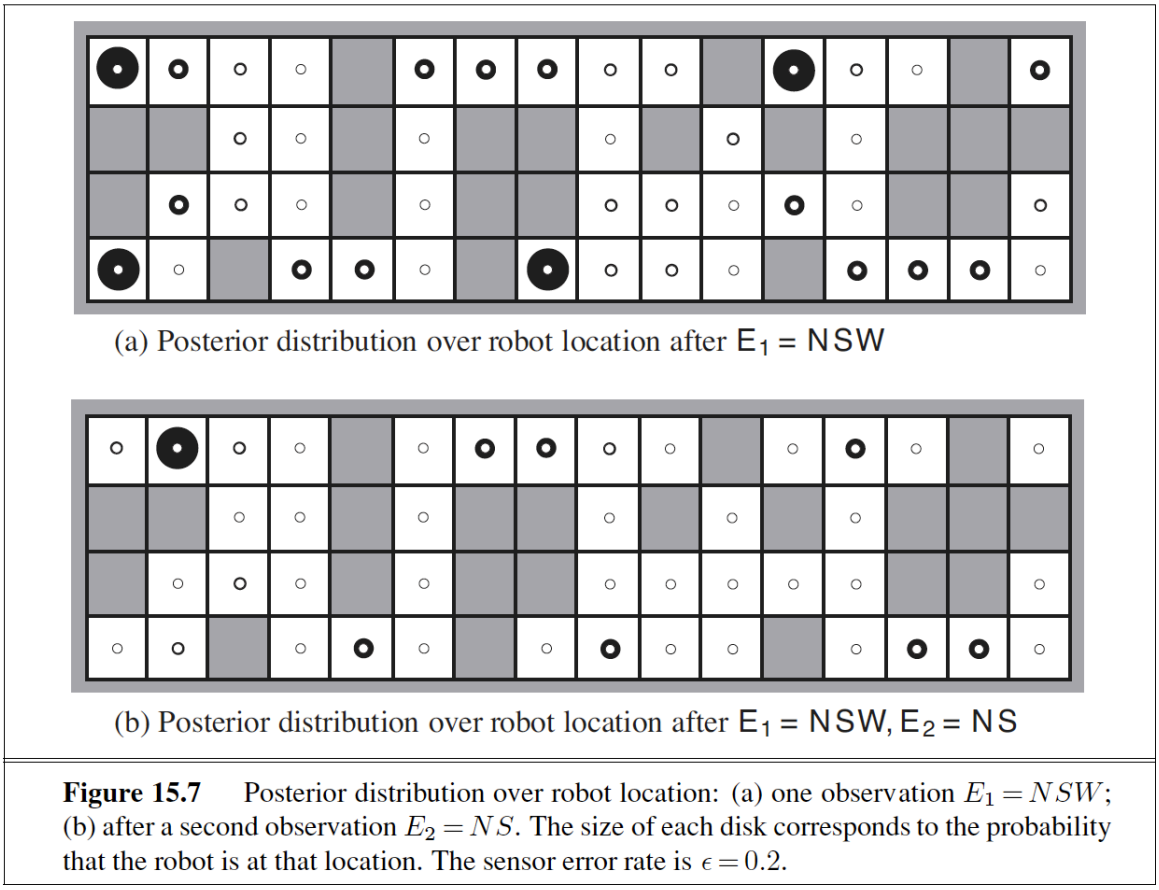

Figure 15.7 Posterior distribution over robot location: (a) one observation $E_1 = NSW$; (b) after a second observation $E_2 = NS$. The size of each disk corresponds to the probability that the robot is at that location. The sensor error rate is $\epsilon = 0.2$.

Figure 2: Figure 15.7 from Russell and Norvig, 3rd Edition, Page 582

Let's consider a more complex robot localization problem. This problem is similar to an example in the Russell and Norvig textbook. In the 3rd edition, check out section 15.3.2, page 582, Figure 15.7. In the 4th edition, check out section 14.3.2, page 477, Figure 14.7.

A robot is in a maze-like environment and the robot has a correct map of the environment. We will use (y, x) or (row, column) to refer to each cell. y is the row value starting with $y = 0$ at the top. x is the column value starting with $x = 0$ at the left.

The robot has one available action called *Move*. Unfortunately, the robot's navigational system is broken. When it executes a *Move* action, it is equally likely to move to any neighboring square. For example, if the robot is in $(0, 2)$, then it moves to $(0, 1)$ with $1/3$ probability, $(0, 3)$ with $1/3$ probability, and $(1, 2)$ with $1/3$ probability.

The robot's task is to determine its current location. To do this, the robot will make use of four noisy sensors that report whether there is an inner or outer wall in each of the four

compass directions. At each time step, the robot receives a sensor reading, which is an integer in $[0, 15]$. If we convert the integer to a four-bit binary number, then each bit gives the presence (bit 1) or absence (bit 0) of an wall in the up, right, down, and left directions. For example, the integer 9 corresponds to the binary number 1001 indicating that there are walls above and to the left of the robot.

Each sensor is noisy and flips the bit with a probability of $0 < \epsilon < 1$. The errors occur independently for the four sensor directions. For instance, suppose that the correct sensor reading is 1001 and the actual sensor reading is 1110. The sensors were correct in one direction and incorrect in three directions. The probability of observing this sensor reading is $(1 - \epsilon)(1 - \epsilon)^3$.

The robot starts at an unknown location. At time step 0, we will assume a uniform distribution over all the squares.

2 Locating the Robot via VEA (71 marks)

You will implement the variable elimination algorithm to perform inference in the complex robot environment introduced in the previous section.

We have provided four python files. Please read the detailed comments in the provided files carefully.

1. `env.py`: Contains a base `Environment` class. **Do not change this file.**
2. `grid_env.py`: Contains a `GridEnvironment` class that extends the base `Environment`, implementing the dynamics from the robot environment. You must implement the `create_trans_probs()` and `create_obs_probs()` methods in the `GridEnvironment` class. **Do not change the function signatures. Do not change anything else in this file!**
3. `factor.py`: Defines a factor class for VEA. **Do not change this file.**
4. `vea.py`: Contains empty functions for VEA operations. You will implement all of the functions in this file. **Do not change the function signatures.**
5. `robot.py`: Provides a demonstration of how to define a `GridEnvironment` object. Feel free to change this file as you desire.

Here are some tips on implementing the variable elimination algorithm.

1. `restrict`: Consider using the splicing operations or `take`.
2. `multiply`: Consider using the `numpy` broadcasting rules to multiply multi-dimensional arrays of different shapes.
3. `sum_out`: Consider using the `sum` operation.

Please complete the following tasks.

1. Implement the empty functions in `grid_env.py` and `vea.py`. Submit these two files only on Marmoset.

   Marking Scheme: (64 marks)

   Unit tests:

- `normalize`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `restrict`
  (1 public test + 2 secret tests) * 2 marks = 6 marks
- `sum_out`
  (1 public test + 2 secret tests) * 2 marks = 6 marks
- `multiply`
  (1 public test + 7 secret tests) * 2 marks = 16 marks
- `ve`
  (1 public test + 5 secret tests) * 3 marks = 18 marks
- `create_trans_probs`
  (1 public test + 2 secret tests) * 2 marks = 6 marks
- `create_obs_probs`
  (1 public test + 2 secret tests) * 3 marks = 9 marks

2. Once you have implemented the functions, let us play with the robot environment. In `robot.py`, we have provided some code to create the robot environment. Add the code for running the variable elimination algorithm, and then run the code to answer the following questions.

   We highly recommend using the provided `visualize_state()` and `visualize_belief` functions to visualize the robot's location and the robot's beliefs.

   (a) If $\epsilon = 0.05$, what is the minimum number of time steps it takes for the robot to determine its current location with a probability of at least 80%?

   Provide an answer and justify your answer with the output for one run of the program. For each time step, the output must specify the state with the largest probability and the corresponding probability. Since the answer may vary between runs, you can run the program for a few times and take the average.

   Marking Scheme: (3 marks)

   - (1 mark) Correct answer.
   - (2 marks) Reasonable justification.

   (b) How does the answer to the question above change if $\epsilon = 0.10$?

   Provide an answer and justify your answer with the output for one run of the program. For each time step, the output must specify the state with the largest probability and the corresponding probability. Since the answer may vary between runs, you can run the program for a few times and take the average.

Marking Scheme: (3 marks)

- (1 mark) Correct answer.
- (2 marks) Reasonable justification.

(c) How does the minimum number of time steps change as a function of ϵ? Describe your conjecture in 1-2 sentences.

Marking Scheme: (1 mark) Reasonable answer