Assignment I: Advanced Robot Navigation (RBE 595)

Pradnya Sushil Shinde

February 3, 2024

1 Bayes Filter

1.1 TASK I

Problem Statement: Analyze the sensor (measurement) model described in Equation 1. What does this tell us about the usefulness or reliability of this sensor? Using the style of notation demonstrated in the lecture notes and Equation 1, write out the mathematical representation of the propagation model: $p(x_t, u_t|x_{t-1})$ for all possible prior states and inputs. What does this model indicate about the system dynamics?

Solution:

By gauging equation 1, we can say that the sensor seems to be less reliable in determining a true positive state (= 0.6). In other words, it will succeed only 6 out of 10 times in sensing that the door is open when the door is open. However, it is more reliable in determining a true negative rate (= 0.8), that is it succeeds 8 out of 10 times. We can say that the system may have been able to perform better if the true positive rate of the sensor was a little higher. Since it is given that the camera used by the robot is black and white and of poor quality, we can say that these rates of determining door states are good given the sensor quality and limitations. Of course, replacing the sensor with a high-quality and better-tuned module will improve the results. But overall, the sensor is quite reliable and useful for mundane tasks such as determining the door's state.

Mathematical Representation of Propagation module:

$p(x_t = \text{Open}, u_t = \text{Push}|x_{t-1} = \text{Open}) = 1$
$p(x_t = \text{Open}, u_t = \text{Push}|x_{t-1} = \text{Close}) = 0.8$
$p(x_t = \text{Close}, u_t = \text{Push}|x_{t-1} = \text{Open}) = 0$
$p(x_t = \text{Close}, u_t = \text{Push}|x_{t-1} = \text{Close}) = 0.2$
$p(x_t = \text{Open}, u_t = \text{do nothing}|x_{t-1} = \text{Open}) = 1$
$p(x_t = \text{Open}, u_t = \text{do nothing}|x_{t-1} = \text{Close}) = 0$
$p(x_t = \text{Close}, u_t = \text{do nothing}|x_{t-1} = \text{Open}) = 0$
$p(x_t = \text{Close}, u_t = \text{do nothing}|x_{t-1} = \text{Close}) = 1$

The above module indicates that there is an imbalance in state estimation rates since the true positive and true negative rates are not equal to each other. This may have a systematic effect on sensor readings and it can induce violations of Markov's assumption which postulates that the past and future data can be independent if one knows the current state. Since there is imbalance and inaccuracy in determining the current state, the model may not be reliable.

1.2 TASK II

Import the necessary library.
Define initial variables.

```
#!/usr/bin/env python

# System State: x_t
# Measurement State: z_t
# Input to the System: u_t

# p(z_t = 1 | x_t = 1) = 0.6
# p(z_t = 0 | x_t = 1) = 0.4
# p(z_t = 0 | x_t = 0) = 0.8
# p(z_t = 1 | x_t = 0) = 0.2

import argparse

robot_state_push = {'OpenOpen':1, 'OpenClose':0.8, 'CloseOpen':0, '
    CloseClose':0.2}
robot_state_do_nothing = {'OpenOpen':1, 'OpenClose':0, 'CloseOpen':0, '
    CloseClose':1}
model_probabilities = [[0.8, 0.4], [0.2, 0.6]]
# To access elements: define {model_probabilities[z_t][x_t]}
```

The prediction step predicts the next stage of the robot based on the current rate and action to be taken.

```
def prediction(action, bel_X0_open, bel_X0_close):
    # Defining a prediction Step that takes in 4 input arguments.
    # bel(X1) = p(X1 | U1 = do nothing, X0 = is open) bel(X0 = is open)
    # + p(x1 | U1 = do nothing, X0 = is closed) bel(X0 = is closed)
    # U1 - Input to the System - Action
    # X1 - Posterior Belief (Close or Open)
    # X0 - Prior Belief (Close or Open)
    # print("Predicting state")
    if action == 'do nothing':
        # bel(X1) = p(X1 = Close| U1 = do nothing, X0 =Open) bel(X0 = is
        # open) + p(x1 = Close | U1 = do nothing, X0 = Close) bel(X0 = is
    closed)
        bel_bar_X1_open =  robot_state_do_nothing['OpenOpen']*bel_X0_open +
     robot_state_do_nothing['OpenClose']*bel_X0_close
        bel_bar_X1_close = robot_state_do_nothing['CloseOpen']*bel_X0_open
    + robot_state_do_nothing['CloseClose']*bel_X0_close
```

```
    elif action == 'push':
        # bel(X1) = p(X1 = Open | U1 = push, X0 = Open) bel(X0 = is open) +
     p(X1 = Open | U1 = push, X0 = Close) bel(X0 = is closed)
        bel_bar_X1_open =  robot_state_push['OpenOpen']*bel_X0_open +
    robot_state_push['OpenClose']*bel_X0_close
        bel_bar_X1_close = robot_state_push['CloseOpen']*bel_X0_open +
    robot_state_push['CloseClose']*bel_X0_close
    else:
        print('No relevant input action!')

    # print("Prediction Step Completed!")

    return bel_bar_X1_open, bel_bar_X1_close
```

The measurement step updates the belief based on the given current state.

```

def measurement(sense, bel_bar_X1_open, bel_bar_X1_close):
    # bel(X1) = eta p(Z1 | X1) bel_bar(X1)
    # X1 - Posterior Belief (Close or Open)
    # Z1 - Sense Measurement
    # eta - Normalization Factor
    # Updating measurement based on predicted belief.
    if sense == 'Open':
        # Calculate for when Robot senses the door to be 'Open'.
        # bel(X1) = eta p(Z1 = sense_open | X1) bel_bar(X1)
        bel_X1_open = model_probabilities[1][1]*bel_bar_X1_open
        bel_X1_close = model_probabilities[1][0]*bel_bar_X1_close

        eta = 1 /(bel_X1_open + bel_X1_close)

        bel_X1_open = eta*bel_X1_open
        bel_X1_close = eta*bel_X1_close

    elif sense == 'Close':
         # Calculate for when Robot senses the door to be 'Close'.
        # bel(X1) = eta p(Z1 = sense_close | X1) bel_bar(X1)
        bel_X1_open = model_probabilities[0][1]*bel_bar_X1_open
        bel_X1_close = model_probabilities[0][0]*bel_bar_X1_close

        eta = 1 /(bel_X1_open + bel_X1_close)

        bel_X1_open = eta*bel_X1_open
        bel_X1_close = eta*bel_X1_close

    # print("Update step compled!")

    return bel_X1_open, bel_X1_close
```

Defining functions to compute the required arguments.

```
def compute_steady_state_belief(bel_open, bel_close, prev_bel_open,
    prev_bel_close, action, sense, threshold):
        while abs(bel_open - prev_bel_open) > threshold or abs(bel_close -
    prev_bel_close) > threshold:
            prev_bel_open, prev_bel_close = bel_open, bel_close
            bel_bar_open, bel_bar_close = prediction(action, bel_open,
    bel_close)
            bel_open, bel_close = measurement(sense, bel_bar_open,
    bel_bar_close)

        return bel_open, bel_close
```

```
def compute_iterations(its, bel_X1_open, bel_X1_close, action, sense):

        while bel_X1_open < 0.9999:
            bel_bar_X1_open, bel_bar_X1_close = prediction(action,
    bel_X1_open, bel_X1_close)
            bel_X1_open, bel_X1_close = measurement(sense, bel_bar_X1_open,
     bel_bar_X1_close)
            its+=1
        return its
```

To answer the following questions, pass the corresponding arguments:

- If the robot always takes the action "do nothing" and always receives the measurement "door open" how many iterations will it take before the robot is at least 99.99% certain the door is open?

  Arguments:
  - - **InputAction** : "do nothing"
  - - **SensedMeasurement** : "Open"
  - - **InitDoorOpenBel** : 0.5

  Output:
  Number of iterations to perform action: push is: 9.

- If the robot always takes the action "push" and always receives the measurement "door open" how many iterations will it take before the robot is at least 99.99% certain the door is open?

  Arguments:
  - - **InputAction** : "push"
  - - **SensedMeasurement** : "Close"
  - - **InitDoorOpenBel** : 0.5

  Output:
  Number of iterations to perform action: push is: 4.

- If the robot always takes the action “push” and always receives the measurement “door closed” what is the steady state belief about the door? Include both the state and the certainty.

  Arguments:
  - - **InputAction** : ”push”
  - - **SensedMeasurement** : ”Close”
  - - **InitDoorOpenBel** : 0.5

  Output:
  Steady state belief for door open is: 0.999955060355391
  Steady state belief for door close is: 4.49396446090398e-05

```
1
2  def main():
3      Parser = argparse.ArgumentParser()
4      Parser.add_argument(
5          "--InputAction",
6          default= 'do nothing',
7          help="Sequence of Input ",
8      )
9      Parser.add_argument(
10         "--SensedMeasurement",
11         default= 'Open',
12         help="Sense whether the door is Open or Close")
13
14     Parser.add_argument(
15         "--InitDoorOpenBel",
16         default= 0.5,
17         help="Initial belief that the door is Open (Optional)",
18     )
19
20     Args = Parser.parse_args()
21     bel_init_open = Args.InitDoorOpenBel   # Provide an initial probability
       that the door is open.
22     action = Args.InputAction # Provide an action to perform
23     sense = Args.SensedMeasurement # Provide the  sensed measurement
24
25     # Given inital belief of door open, calcualte initial belief of door
       close
26     bel_init_close = 1 - bel_init_open
27
28     # Set iterations to zero
29     its = 0
30     bel_X1_open =  bel_init_open
31     bel_X1_close = bel_init_close
32
33     print("Computing iterations to perform action")
```

```
    its = compute_iterations(its, bel_X1_open, bel_X1_close, action, sense)
    print("Number of iterations to perform action: " + action + " is: " +
    str(its))

    bel_open, bel_close = bel_init_open, bel_init_close
    prev_bel_open, prev_bel_close = 0, 0
    threshold = 1e-4

    print("Computing steady state belief.")

    open_steady_state_bel, close_steady_state_bel =
    compute_steady_state_belief(bel_open, bel_close, prev_bel_open,
    prev_bel_close, action='push', sense='Close', threshold=threshold)
    print("Steady stae belief for door open is: "+ str(
    open_steady_state_bel))
    print("Steady stae belief for door close is: "+ str(
    close_steady_state_bel))

#Calling the main() function
main()
```