CS 3630 Project 2

Name: Fei Ding
GT email: fding33@gatech.edu
GT username: fding33
GTID: 903444656

1. What is the difference between CDF and PMF?

PMF stands for probability mass function while CDF stands for cumulative distribution function. PMF tells you the exact probability of being in one state, and all entries sum up to 1. On the other hand, CDF tells you the cumulative probability of being either in one state or the states before it, supposing these states have been sorted in linearly ordered list. To put in different words, if we are dealing with discrete states then PMF is like performing a cumulative sum on the PMF function, and if we have continuous states then CDF is the integral of PMF by definition. In particular, after calculating CDF from PMF, it becomes extremely handy to perform inverse transform sampling in robotics simulations.

2. How did you implement maximum-probable explanation?

We first convert the DBN into a factor graph, where each unknown variable is a factor and their connections are abstracted into functions, denoted as ϕ's. Then, we define a function ψ for each paired unknown states as the product of ϕ's it connects to. We will then try to iteratively eliminate each state variable from X_1 to X_n , by assuming the value of X_{i+1} and test out $\text{the}\, X_i$ that maximizes its corresponding ψ. In this way we are creating a chained lookup tables each of which tells the most probable previous state given the current state, and the maximized values as a function τ used in next iteration. We can finally use back-substitution method to generate the entire most probable states chain from the last state. This is called max-product algorithm for HMM.

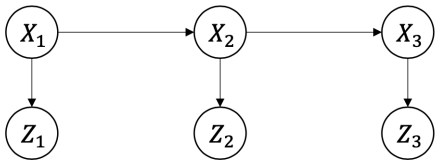
3. What is the purpose of sampling and how is it used?

Sampling is used to simulate robot sensors and actions to account for uncertainty by picking a random value from PMF (in real computing we first calculate CDF and then do inverse transform sampling). In statistics, sometimes it is unnecessary to get a precise result as census, so sampling can be used for both saving time and getting fairly accurate inferential results if the sampling is done properly with a reasonably size. In the context of DBN, if we know the whole network structure and has the conditional probability tables for each connection, we can simulate the flow of the network using sampling methods. Sampling is critical because in the field of robotics uncertainty/stochasticity is everywhere and ignoring them usually leads to far-off results.

4. What is a factor graph?

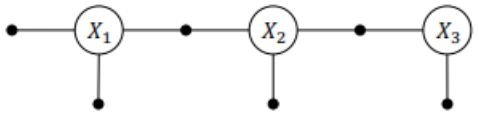
A factor graph is a bipartite graph model of random variables and factors. Factors describe the dependencies and relations of the variables. It represents a similar idea to Bayes net and in fact a Bayes net can easily equate to a factor graph by converting each arrow into a factor if that factor relates to an unknown variable in the Bayes net. Each factor is a function of variables it connects to, and by abstracting the relations into functions we analyze the network more easily: isolating each variable with its factors and trying to eliminate them, which useful for inference algorithms like max-product algorithm and max-sum algorithm.

DBN (HMM)



Observations known,
States unknown

Factor Graph



5. What did you learn in this project?

I learnt a little bit of NumPy as it is very useful in creating CDF in this project. The idea of functional programming becomes essential in this project because it provides more abstraction, and a lot of algorithms like sampling in DBN and implementing MPE are easier to implement as I need to write fewer lines of code. The project also helped me learnt a great amount of useful techniques and concepts used in robotics by coding them real. I implemented things like sensor models and transition models and used them simulate Bayes net using sampling methods. I also learnt to relations between Bayes nets and factor graphs and how the later can help me do inference with highly efficient and specialized algorithms. Now I can really see how these littles pieces connect and work out for robotics!

6. Screenshot and paste the output of running your unit tests here

```
def suite():
  functions = [
               'test_functions_exist',
               'test_actions_exist',
               'test_action_prior',
               'test_states_exist',
               'test_state_prior',
               'test_calculate_cdf',
               'pmf_action_sanity_check',
               'test_sensor_model',
               'test_transition_model_center_normal',
               'test_transition_model_corner_wall',
               'test_sample_from_dbn_types',
               'test_maximum_probable_explanation',
               'test_sample_sensor_model',
               'test_sample_transition_model'
               ]
  suite = unittest.TestSuite()
  for func in functions:
    suite.addTest(TestFunctions(func))
  return suite

if __name__ == '__main__':
    runner = unittest.TextTestRunner()
    runner.run(suite())
```

```
..............
----------------------------------------------------------------------
Ran 14 tests in 1.098s

OK
```

7. Extra-Credit - MPE: Screenshot and paste the output of running your extra-credit unit tests here

Extra Credit (General MPE) Test Case

Run this to check if your `general_maximum_probable_explanation` passes the given test case. Make sure to add the screenshot of the results in the appropriate slide in the reflection.

```
[ ] class TestMPE(unittest.TestCase):
      def test_general_maximum_probable_explanation(self) -> None:
          """
          Checks if general_maximum_probable_explanation is working properly.
          """
          actions = ['Down', 'Down', 'Right', 'Up', 'Up', 'Left']
          observations = [3, 4, 5, 5, 4, 3, 3]
          correct_states = [(3,3), (4,3), (5,3), (5,4), (4,4), (3,4), (3,3)]
          assert general_maximum_probable_explanation(actions, observations) == correct_states,\
              "Correct state does not return correctly"

    if __name__ == '__main__':
      suite = unittest.TestSuite()
      suite.addTest(TestMPE("test_general_maximum_probable_explanation"))
      runner = unittest.TextTestRunner()
      runner.run(suite)
```

```
.
----------------------------------------------------------------------
Ran 1 test in 0.659s

OK
```

8. Extra-Credit - Portal: Screenshot and paste the output of running your extra-credit unit tests here

```
[ ]  class TestPortal(unittest.TestCase):
       def test_transition_model_portal(self) -> None:
           """
           EXTRA CREDIT
           Checks if the transition model is working properly for the portal case.
           """
           curr_state = (3,4)
           curr_action = 'Right'
           correct_state = (6,3)
           incorrect_states = [(3,3), (2,4), (4,4)]
           invalid_states = [(0,0), (1,1), (2,2)]
           assert transition_model_portal(correct_state, curr_state, curr_action) == 0.85,\
               "Correct state does not return 0.85"
           for curr_incorrect_state in incorrect_states:
               assert transition_model_portal(curr_incorrect_state, curr_state, curr_action) == 0.05, \
                   "Incorrect state does not return 0.05"
           for curr_invalid_state in invalid_states:
               assert transition_model_portal(curr_invalid_state, curr_state, curr_action) == 0, \
                   "Invalid state does not return 0"

     if __name__ == '__main__':
       suite = unittest.TestSuite()
       suite.addTest(TestPortal("test_transition_model_portal"))
       runner = unittest.TextTestRunner()
       runner.run(suite)
```

```
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```