Project 2

Section 3.1 State Abstractions

1) MCQs to identify Bayesian vs. Frequentist

- Finding probability of getting a one after rolling dice 1000 times

Frequentist would repeat running a simulation of 1001 dice roles and then look at the average for the 1001st dice role. A bayesian approach would to be to use the past 1000 dice roles to decide the probability of the 1001st dice role.

- Finding probability of getting A in CS 3630 next semester

A frequentist would let thousands of students take the course and calculate the average percentage of A earning students. A bayesian approach would be to look at how the student is doing in the semester and calculate the probability based off of the trends in their grades.

- Finding the probability of Rafael Nadal winning the 21 grand slam titles

A frequentist would run a simulation to see how many baseball players win the 21 grand slam titles and then use the rarity as the estimation. A bayesian approach is to look at Rafael Nadal’s track record and use it to calculate the probability

2) What will be the discrete distribution if the robot ALWAYS starts out in the Living Room?

```
vacuum.rooms = ["Living Room", "Kitchen", "Office", "Hallway", "Dining Room"]
X = VARIABLES.discrete_series("X", [1], vacuum.rooms)
prior = gtsam.DiscreteDistribution(X[1], "1/0/0/0/0")
pretty(prior)
```

X1	value
Living Room	1
Kitchen	0
Office	0
Hallway	0
Dining Room	0

If robots always starts out in the living room, and since the prior distribution will only encode knowledge about the initial state of the robot, we could build the following probability distributiont able:

X1	value
Living Room	1
Kitchen	0
Office	0
Hallway	0

Dining Room	0

Section 3.2 Vacuum Action

1. Even though the CPT in section 3.2.1 in the book above has 100 numbers in it, how many independent degrees of freedom do we actually have when specifying this CPT?

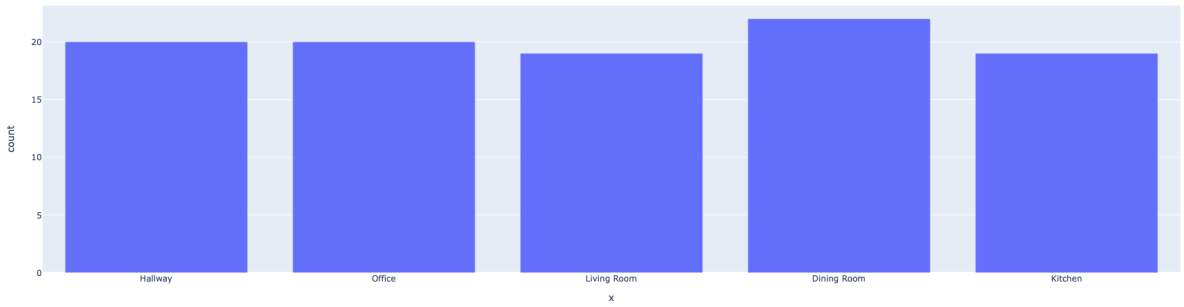
30 (legal moves)

2. If we start from the Dining Room and attempt to go left, i.e., action A0 = L, what is the PMF over the next state?

Dining Room	L	0	0	0	0.8	0.2

Section 3.3 Vacuum Sensing

1. Please run the ancestral sampling 100 times using the prior given by the TAs. Plot the histogram of the starting state X1.



2. We see that we are sometimes starting in the living room when we are running the ancestral sampling algorithm. What would you change to always start in the office?

I would change the prior to “0/0/1/0/0” to allow the probability to be certain in the favor of the office.

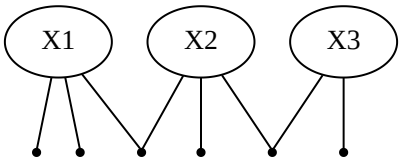
Section 3.4 Vacuum Perception

1. Calculate the size of the table needed to enumerate the posterior over the states S the robot dynamic Bayes net from the previous section, given the value of all observations O and actions A.

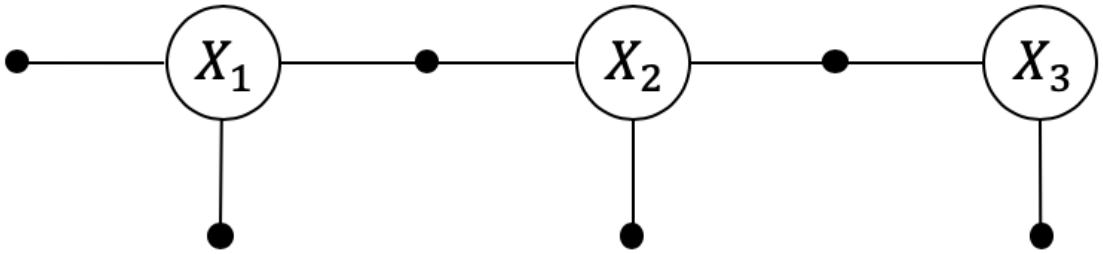
2^5=32

2. Given the DBN from the textbook,

- If the actions and the measurements are known, how does the factor graph look like?

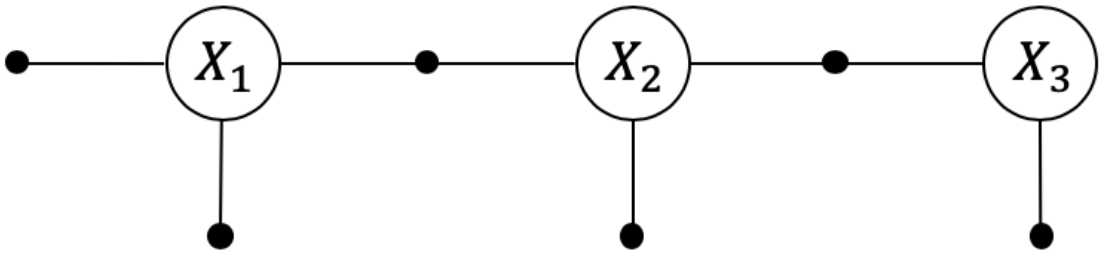


- If the actions and the measurements are unknown, how does the factor graph look like?

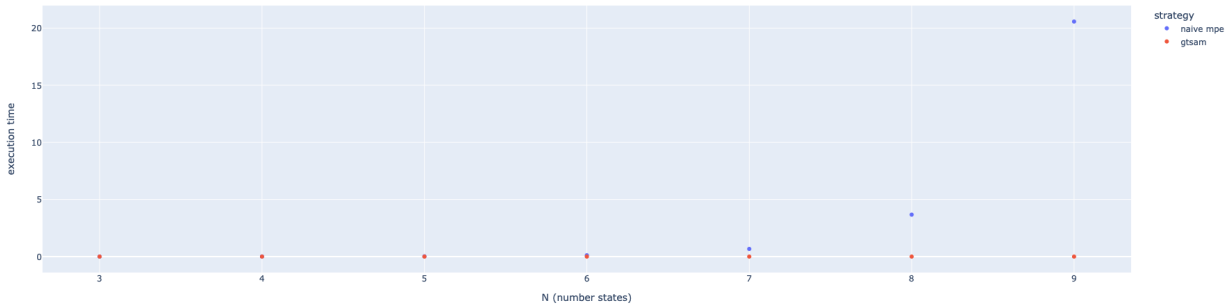


- If the actions and the measurements are unknown, but the trajectory of the states is known, how does the factor graph look like? Do you observe any direct findings from the factor graph?

You understand the general direction of the graph.



3. Please plot the graph from the coding section (3.4) which compares the time complexity of Naïve MPE implementations vs GTSAM implementation



4. What is the time complexity of MPE when enumerating over an N different number of states? Choose from the option

   ● Linear(ax+c) ● Cubic(ax3+c) ● Quintic(ax5+c) **● Exponential(ex+c)**

   Exponential

Section 3.5 Vacuum Decision

1. What is roll-out reward?

The expected reward value given an input rollout. This reward value is used as quantitative criteria for evaluating actions and their effects. It is calculated using a reward function.

2. Compare two control tapes provided in the code. What is the optimal control tapes? Why?

The second one contains more ups thus putting it at a higher likelihood to end up in the living room.

```
def Average(lst):
    return sum(lst) / len(lst)

print(Average([control_tape_reward(markovChain, "Office", actions_1,R,A,X) for i in range(1000000)]))
print(Average([control_tape_reward(markovChain, "Office", actions_2,R,A,X) for i in range(1000000)]))
```

The reward function only applies to a specific instance of the case so the results can vary. I ran both tapes 1000000 times and found the average of the trials for each one. The results were:

3.376493 for {A[1]:"R", A[2]:"U", A[3]:"R", A[4]:"U"}
3.60111 for {A[1]:"R", A[2]:"U", A[3]:"U", A[4]:"U"}

Due to the law of large numbers we can see the second one averages a higher reward thus its more optimal.

3. Based on the code(Policy Iteration function) you ran, what is the optimal policy?

```
def Average(lst):
    return sum(lst) / len(lst)

print(Average([control_tape_reward(markovChain, "Office", actions_1,R,A,X) for i in range(1000000)]))
print(Average([control_tape_reward(markovChain, "Office", actions_2,R,A,X) for i in range(1000000)]))
```

The reward function only applies to a specific instance of the case so the results can vary. I ran both tapes 1000000 times and found the average of the trials for each one. The results were:

3.376493 for {A[1]:"R", A[2]:"U", A[3]:"R", A[4]:"U"}
3.60111 for {A[1]:"R", A[2]:"U", A[3]:"U", A[4]:"U"}

Due to the law of large numbers we can see the second one averages a higher reward thus its more optimal.

4. What's the objective of using policy iteration? What's the objective of using value iteration?

Policy iteration improves an initial guess iteratively until no further improvements are possible. Value iteration constructs an array of estimates that eventually converges to V* in nonfinite time. Both’s goal is to find the most accurate prediction of the optimal policy.