Sujay Jakka

Svj0007@auburn.edu

COMP 5600

April 5, 2025

Assignment 3 Report

In this report, I will detail my algorithmic and design choices for Assignment 3. This report will consist of 9 sections, each dedicated to explaining my thought process for each question of the assignment. Furthermore, most of everything that is said in this report is also commented in my code to prevent any confusion.

Question 1

For question 1, we are asked to construct a bayes net with a specific structure where Ghost 0 and Pacman are the parents of Obs 0 and Pacman and Ghost 1 are the parents of Obs 1. To build this bayes net we need to list all the variables in the bayes net, the values these variables can take on, and the edges between these variables. This is pretty trivial. I included all possible positions on the grid, which is the domain for the Pacman, Ghost 1, and Ghost 2. Then I set the domain for the Obs 0 and Obs 1 to be the range of 0 to x range – 1 + y range – 1 + max noise. We subtract -1 to each of the ranges because suppose if our range is 0 indexed from 0 to 6, we have a range of 7 however the max possible distance between two number in this range is 6 – 0 which is 6. This is why we need to subract 1 from each range. Now we loop through 0 to our max range add each of these distance values to the list of domain values for observation 0 and observation 1. Lastly, we add our edges as a list of tuples where the first element in the tuple is the parent and the second element is the child. This was pretty trivial as well. Afterwards we construct a bayes net object and finally return it.

Question 2

For question 2, we are asked to implement the joinFactors function which just allows us to join two factors together for example if we have factors P(X | Y) and P(Y), we want to simplify this to P(X ,Y) through product rule. Furthermore, we are also told that a variable can only appear as an unconditioned variable for only one factor. For this question I just created two sets that stored all the unconditioned variables and all the conditioned variables. Then I made sure to remove any conditioned variables from the conditioned variables set if they are also in the unconditioned variables set as that means after the join this variable will no longer be conditioned. Next, I created a factor object using the unconditioned set, the conditioned set, and one of the factors variable domain dictionary. All factors in this bayes net have the same variable domain dictionary so I was able to pick an arbitrary factor's domain dictionary. After I created that resulting factor object, I looped through every possible assignment in that factor object and came up with the probability by multiplying all of the input factor's probability given that specific assignment. Then I would set that assignment's probability in the factor with .setProbability. Lastly, I would return the resulting factor.

Question 3

For question 3, we were asked to eliminate a certain variable from a factor and then return the resulting factor. This means to sum all the entries in the factor which only differ in the variable being eliminated. One thing to note here is that the elimination variable must be an unconditioned variable in the input factor. For this question, I created two sets again one for unconditioned variables and one for conditioned variables. I then added all of the input factors unconditioned variables to this unconditioned variables set and all of the input factors conditioned variables to the conditioned variables set. I then made sure to remove the elimination variable from the unconditioned variable set as the resulting factor will have this variable eliminated. I then created a new factor object from these two sets and the input factor’s variable domain dictionary. I then looped through each possible assignment in the resulting factor and summed up all of the entries in the input factor that had all of the same values but only differed in the value of the elimination variable. I then set this resulting probability as the assignments probability. After this is done for every assignment, we return the resulting factor.

Question 4

For question 4, we are asked to implement the inferenceByVariableElimination function. The first thing I did in this function is to only get the factors and their corresponding entries that match the evidence given in the function. We are required to eliminate the variables in a specific order. I loop through each elimination variable and call the function joinFactorsByVariable that allows you to input factors and their elimination variable and it will join only the factors that contain that elimination variable. It will then return a joined factor and also a list of factors that were not joined. I set this list of factors that were not joined to be my new list of factors. Next, I check if this joined factor has only one unconditioned variable if it does then there is no reason to do elimination on this factor as when you do elimination it will just result in 1 for all the entries which is not a valid factor, and it will also have no impact on future joins. If there is not only one unconditioned variable in the joined factor, then we call the eliminate function. Lastly at the end of the loop, we add this resulting factor to our list of factors. Once we are outside the loop and eliminated all of the elimination variables, we join all the factors once more and then normalize. We then return this final normalized factor.

Question 5a

For question 5a, we were asked to implement the normalize and sample functions. For the normalize method I just summed all of the probability values of the distribution. Next, for each probability value I divided this by the total sum to normalize. For the sample method, I used python’s random package which has a function called random.choices which allows you to input a list of elements to select and their corresponding weight, and it will perform a weighted sampling. I used this function and used the distribution’s keys as elements to select from and the probability values as the weights for each element. I then sampled and returned a single element.

Question 5b

For question 5b, we were asked to implement the getObservationProb function. This function basically asks us to find P(noisy distance | true distance) given a noisy distance to ghost, pacman’s position, and the ghost's position. In the beginning of the function, I first implement some of the edge cases. If the ghost position is equal to the jail position, that means the ghost is in jail. This should mean the noisy distance should be none and should have a probability of a 100% of saying none. If we are given a noisy distance that is anything else but none when the ghost is in jail, we return 0. If it is none, we return 1. For the condition where the ghost is not in jail and the noisy distance is none, 0 should be returned as there is a 0% chance of this happening. Now if the ghost is not in jail and the noisy distance is not none, we find the true distance by calling the manhattan distance function to find the manhattan distance between pacman and the ghost. We then call the busters.getObservationProbability function with the noisy and true distance as input to find P(noisy distance | true distance). We then return this probability.

Question 6

For question 6, we are implementing the observeUpdate function to update self.beliefs(our prior probability) which is represented as P(Ghost Position | All previous observations) to P(Ghost Position | All Observations). We are given likelihood in the form of P(New Observation | Ghost Position) which can be used to update our prior. For this function I looped through each position and found the corresponding prior probability for each position in self.beliefs. I then multiplied our likelihood, which can be found by calling the getObservationProb function with our observation, gamestate, ghost position, and jail position and input, to our prior and then set the corresponding probability of this position to this new posterior probability.

Question 7

For question 7, we are asked to implement the elapseTime function which essentially wants us to find the probability distribution of the ghost position at time t + 1. For this function, I first created a new discrete distribution object called new beliefs. Then I looped through every position, and set this position as the old position(time t position) of the ghost. I then found the probability distribution that the ghost will be at a certain position given this old position by calling the function getPositionDistribution with the game state and old position as input. Lastly I looped through each position in this P(Ghost Position t + 1 | Ghost Position t) distribution and then added the product of P(Ghost Position t + 1 | Ghost Position t) with P(Ghost Position t) to the corresponding probability of the new position in my new beliefs distribution. After doing this for every old position, I returned my new distribution which reflects P(Ghost Position t + 1).

Question 8

For question 8, we are implementing the chooseAction function that finds the action that yields the minimum distance to a most probable ghost position. I first create a list called most likely positions which is to store the most likely position of each ghost according to each ghosts position probability distribution. I then loop through each ghost's probability distribution to find this most likely position for each ghost, and I append it my most likely positions list. Next, I loop through all possible actions for pacman at the given position and for each action I find the position that yields from that action. I then find the distance between this position and each ghosts most likely position and if this action yields the smallest distance to a ghost position I update what is considered the best action and I also update the minimum action distance. Lastly, after I finish looping through all actions, I return the best action which is the overall action that yields the smallest distance to a ghost position.