Due Date: November 8 11:59pm

Submission Instructions: Assignments are to be submitted through LEARN, in the Dropbox labelled *Assignment 3 Submissions* in the Assignment 3 folder. Late assignments will be accepted up until November 10 at 11:59pm. Please read the course policy on assignments submitted after the official due date. *No assignment will be accepted, for any reason, after 11:59pm on November 10.*

Lead TA: Ankit Vadehra (avadehra@uwaterloo.ca) Office hours on Fridays 10:30am-12:00pm in DC 2517.

Announcements: The following exercises are to be done individually. For Question 2 you may use the programming language of your choice.

Question 1

Every morning you encounter the same problem – your robot never manages to dress itself properly! It puts is socks on its hands, uses its sweater as shoes, and wears its pants like a hat. If you dress the robot yourself, then you have less time for your coffee, which is an unacceptable outcome. Thus, you decide it is time for your robot to learn how to dress itself. In this question you will use *Q-Learning* so that your robot learns how to put on its clothes properly.

There are 4 articles of clothing:

- Shirt
- Sweater
- Socks
- Shoes.

There are 3 possible locations for a piece of clothing:

- **R:** The room
- **U:** Upper Body
- **F:** Feet.

At any given state, the robot has the following possible legal deterministic actions:

- If it is wearing at least one piece of clothing, then it can take **1** piece off. This results in the piece of clothing being in the room **R**. However, there are some constraints
  - Shoes must be removed before Socks.

  - The Sweater must be removed before the Shirt.

- The robot can put on an item of clothing on either its Upper Body or Feet.
  - Socks must be put on before Shoes.
  - The Shirt must be put on before the Sweater.

Furthermore, you can assume that Socks and Shoes go on the Feet and the Sweater and Shirt go on the Upper Body. (This makes the final state space manageable!) Illeagal actions are also possible, and would involve violating one of the constraints listed above. For example, illegal actions would be putting Shoes on the Upper Body or the Sweather on before the Shirt.

State Representation: A state is defined as a 4 character string, where each character represents the location of a particular piece of clothing. The first character represents the location of the shirt, the second character represents the location of the sweater, the third character represents the location of the socks, and the fourth character represents the location of the shoes. For example, *RRRR* represents the state where all clothes are in the room (i.e. not on the robot), *URRR* represents the state where the robot is wearing only the Shirt. The goal state is *UUFF* which is the state where the robot is wearing the shirt and sweater on its Upper Body and the Socks and Shoes on its Feet.

a. (5 points) Draw the transition diagram for the problem. A transition diagram contains a node for each state in the state space. A directed edge goes from node A to node B if there is a *legal action* that could be taken in the state corresponding to node A that results in the robot transitioning to the state corresponding to node B.

b. (55 points) Use Q-learning so that your robot learns how to dress itself. We are providing you with partial code in the file *assign3QL.py* as a starting point. You need to implement the following functions:

- **buildTransitionDiag(states):** Build the transition diagram that you created in part *a.*
- **buildRMatrix(tD):** Build the Reward matrix. Transitions that result in the goal state *UUFF* have reward +100. All other legal transitions are assigned reward 0. Illegal transitions have reward -1.
- **learn_Q(R, gamma, alpha, numEpisodes):** This is the main function of the learning algorithm. Include your code to implement Q-learning here.

Please read the documentation provided in *assign3QL.py* to learn more about the functions, and what they must return.

Once you have added your own code to the file, you will train your robot. We provide code for training, for plotting the number of steps taken by the robot to dress itself after different training epsidodes, and to extract the final policy the robot learns.

Run Q-learning for 200 training episodes for $\alpha = 0.1$, 0.5, 0.8, and 1.0. Keep $\gamma = 0.8$ for each run. We are providing code that handles the number of episodes for you. (See documentation). For each value of α generate a graph showing training episode vs. number of moves in the policy. (Note that we are providing the code that will generate the graph for you. You are not required to use this code.).

Submit the following:

- *assign3QL.py* with your code added to it. (20 points)
- The graphs generated for the 4 different values of α. (10 points)
- A discussion of your results. In particular explain how the number of actions taken by the robot to get dressed changes as a function of the number of training episodes. Also explain how the learning rate, α, changes the learning process. (25 points)

Question 2. Variable Elimination (0 points)

This part of the assignment is worth zero points. However, your implementation will be used in the rest of the assignment. *If you do the implementation then upload a copy of it with your assignment.*

Implement the variable elimination algorithm by coding the following four functions in the programming language of your choice.

a. restrictedFactor=restrict(factor, variable, value): a function that restricts a variable to some value in a given factor

b. productFactor = multiply(factor1, factor2): a function that multiplies two factors

c. resultFactor = sumout(factor, variable): a function that sums out a variable given a factor

d. resultFactor = inference(factorList, queryVariables, orderedListOfHiddenVariables, evidenceList): a function that computes *Pr(queryVariable | evidenceList)* by variable elimination. This function should restrict the factors in *factorList* according to the evidence in *evidenceList*. Next, it should sum out the hidden variables from the product of the factors in *factorList*. The variables should be summed out in the order given in *orderedListOfHiddenVariables*. Finally, the answer should be normalized. Note that you might want to implement an additional function **normalize(factor)** to help you do this.

Here are some useful tips for this part of the assignment.

Tip Factors are essentially multi-dimensional arrays. Therefore, you may want to use this data structure. However, you are free to use any data structure that you want.

Tip Test each function individually using the simple examples we covered in class. Debugging the entire variable elimination algorithm at once can be tricky.

Hand in your code. While there are no marks for the code, in Question 3 part marks will be given for using your variable elimination implementation to do the calculations.

Question 3. Bayes Nets (80 points)

Your dog Fido has been howling for the last three hours and you want to decide whether or not to take him to the vet, or just put in earplugs and go back to sleep. You have the following information available to you:

- You know that Fido often howls if he is genuinely sick. However, he is a healthy dog and is only sick 5% of the time. If Fido is really sick he probably has not eaten much of his dinner and has food left in his bowl. In the past you have observed that when Fido is sick then 60% of the time he does not eat. However, about 10% of the time, when Fido is healthy he still does not eat his dinner.

- You also know that Fido often howls if there is a full moon or the neighbour's dog howls. The neighbour's dog sometimes howls at the full moon and sometimes howls when your neighbour is away. However, the neighbour's dog is never affected by Fido's howls. You know that (since you live on Earth) there is a full moon once out of every twenty-eight days. You have observed that your neighbour travels three days out of every ten. You also know that if there is no full moon and the neighbour is at home then their dog never howls, but if there is no full moon and the neighbour is away then the dog howls with probability 0.5. If there is a full moon then the dog is more likely to howl; if the neighbour is at home then it howls with probability 0.4, but if the neighbour is also away then the probability of howling increases to 0.8.

- Finally, you know that if all the triggers are there (i.e. full moon, howling neighbour's dog, Fido sick) then Fido will howl with probability 0.99, but if none of the triggers are there, then Fido will not howl. If Fido is sick then he is likely to howl. In particular, if he is sick (but there are no other triggers) then Fido will howl with probability 0.5. If Fido is sick and there is another trigger then he is even more likely to howl – if the neighbour's dog is also howling then Fido will howl $\frac{3}{4}$'s of the time, while if there is a full moon then Fido howls 90% of the time. If Fido is *not* sick then he is less likely to howl. The full moon and the neighbour's dog will only cause him to howl with probability 0.65, while if there is only a full moon then he will howl 40% of the time, and if there is no full moon, but the neighbour's dog is making noise, then he howl's with probability 0.2.

a. (32 points) Given the information about Fido, construct a Bayes Network. Show the graph and the conditional probability tables. The network should encode all the information stated above. It should contain six nodes corresponding to the following binary random variables:

- *FH* – Fido howls
- *FS* - Fido is sick
- *FB* - there is food left in Fido's food bowl
- *FM* - there is a full moom
- *NA* - the neighbour is away
- *NDG* - the neighbour's dog howls

The edges in your Bayes Network should accurately capture the probabilistic dependencies between these variables.

For the next set of questions indicate what queries (i.e. Pr(vars | evidence)) you used to compute the probability. Whether you answer the queries by hand or by using your code, provide a printout of the factors computed at each step of variable elimination. If you think that some of the probabilities will not change, then you do not have to redo the calculations, but you do need to provide an explanation. Note two things

- The information is cumulative. For example, if the moon is full in Question *b.* then it remains full in Question *c.* etc.
- **Note that a maximum of two thirds of the marks are earned if you answer correctly the question by doing the computations by hand instead of using your program.**

b. (12 points) What is the prior probability that Fido will howl (i.e. Pr(FH))?

c. (12 points) You can hear Fido howling, and you are a little concerned that he is sick. You look out the window and see that the moon is full. What is probability that Fido is sick?

d. (12 points) You next walk to the kitchen to see if there is any food left in Fido's bowl. You note that the bowl is full – that is Fido has not eaten. How does your belief change about Fido being sick now that you know there is a full moon and Fido has not eaten? That is, what is the probability that Fido is sick given the new information?

e. (12 points) Finally, you decide to call your neighbour to see if they are home or not. The phone rings and rings so you conclude that your neighbour is away. Now, given this information, how does your belief about Fido being sick change? That is, what is the probability that Fido is sick given the new information?