CS512– Artificial Intelligence

Instructor : Shashi Shekhar Jha (shashi@iitrpr.ac.in)

Lab Assignment - 2 | Due on 17/04/2021 2400 Hrs (100 Marks)

Submission Instructions:

All submission is through google classroom in one zip file. In case you face any trouble with the submission, please contact the TAs:

- Namrata Lodhi, 2019aim1007@iitrpr.ac.in
- Surbhi Madan, surbhi.19csz0011@iitrpr.ac.in
- Armaan Garg, 2019csz0002@iitrpr.ac.in

Your submission must be your original work. Do not indulge in any kind of plagiarism or copying. Abide by the honour and integrity code to do your assignment.

As mentioned in the class, late submissions will attract penalties.

Penalty Policy: There will be a penalty of 5% for every 24 Hr delay in the submission. E.g. For 1st 24 Hr delay the penalty will be 5%, for submission with a delay of >24 Hr and < 48 Hr, the penalty will be 10% and so on.

You submission must include:

- A legible PDF document with all your answers to the assignment problems, stating the reasoning and output.
  Note: PDF file is mandatory to get the assignment evaluated

- A folder named as 'code' containing the scripts for the assignment along with the other necessary files to run yourcode.

- A README file explaining how to execute your code.

Naming Convention:
Name the ZIP file submission as follows:
YourName_rollnumber_Assignmentnumber.zip

E.g. if your name is ABC, roll number is 2020csx1234 and submission is for lab2 then you should name the zip file as: ABC_2020csx1234_lab2.zip

PART I

Gibbs Sampler [Total 20 points]

Assume we are interested in simulating from the bivariate Gaussian distribution $N_2(\mu, \Sigma)$ where

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & a \\ a & 1 \end{pmatrix}$$

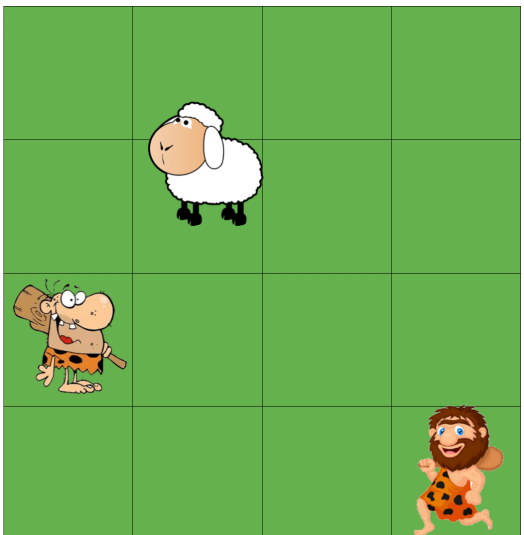
A. Derive the conditional distributions for x1 given x2 (i.e. p(x1/x2)) and x2 given x1 (i.e. p(x2/x1)). [5 **points**]
B. Implement the Gibbs sampler based on the conditional distributions. [5 **points**]
C. Run the algorithm for 10,000 iterations for a = 0. Use simulations from the standard Gaussian distribution as starting points. Estimate μ and Σ from your simulations. Use traceplots to see how many of the first samples you should discard. Make a plot of your simulations in the two-dimensional space plotting both x1 & x2, drawing a line between each iteration. Also plot the samples taken directly from the bivariate Gaussian distribution $N_2(\mu, \Sigma)$. Comment on the results. [5 **points**]
D. Now repeat the previous sampling with a = 0.99. Make similar estimates and plots and comment on the results. [5 **points**]

PART II

Prey and Predator model using Bayesian Network

(Total 20 points)

In this problem, two cavemen from Stone-age are trying to capture a sheep. Obviously the sheep can run faster than the cavemen. The cavemen are starving and if they do not get the sheep within a certain amount of time (turns) they may die out of hunger.



The strategy to capture the sheep involves predicting where the sheep is going to flee to or try to lure it into a corner.

The rules for the game can be seen in the list below:

1. The environment is a board/matrix with each tile defining a position
2. As mentioned, there are two types of players on the board (cavemen & sheep)
3. At most one player can be positioned on each tile
4. When a caveman is placed on a tile next to the sheep, the sheep is captured by the caveman and the game ends.
5. The sheep can move at most two tiles at a time, but only diagonally with both steps in the same direction.
6. The cavemen can move one tile at the time in only vertical and horizontal directions.
7. The sheep cannot be captured diagonally by the caveman
8. A player can choose not to move during a turn
9. All players make their move at the same time

Note: Consider board size is of 8*8 for reasonable run time during implementation.

Below are the four parts of your assignment that need to be completed. Design these scenarios using matrix and observe the different outputs.

[Part 1: 5 marks] – A sheep that works with a simple deterministic algorithm, that tries to evade the cavemen and avoid getting trapped in a corner. – Two cavemen that work with a simple deterministic algorithm that always tries to follow and capture the sheep – The board has no obstacles. – All positions of other players are available to all others. Play 2 games, upto 100 movements and output the agent that wins.

[Part 2: 15 marks] – One of the cavemen (caveman 1) uses a Bayesian Network (see Fig.1.) that tries to predict the sheep's next move and thereby *learn the sheep's strategy (cavemen otherwise have no access to the sheep's algorithm of movement in the grid)*. Implement the Bayesian caveman using the information in the Help below. Test the system by playing 5 games, upto 200 movements and output the agent that wins.

Help--
Caveman's Simple Algorithm

```
For each possible move
  Calculate the distance to sheep
Select shortest distance to sheep after move, and execute.
```

Sheep's Algorithm

```
IF there is no cavemen within radius (2 moves of a caveman)
       do not move
       ELSE for each possible move
        calculate the sum of: distance to nearest corner,
        distance to caveman 1 and distance to caveman 2,
          for each possible move.
            Choose the move with the highest sum.
```

Implementation of the game

The main idea is that we have a list of cavemen, each with an ID, and a list of sheep who also have an ID. These are used to perform the placement of the players on the board, and once this is done each of the players are asked to perform a choice as to where they want to move. Once all players have given their answer the players are moved to their target, and board configuration is tested for whether it is a winning configuration or not. If the game is not won the players are once again asked to perform an action, and so on.

Note: If the game has not been won in 200 turns the game should be terminated (as the game might have entered a loop).

Bayesian Network Setup

For coupling the Bayesian Network together with the caveman agent in the code, a set of input and output is required. The input for the Bayesian Network is given by the agent code, and the Bayesian Network returns the new probabilities. These are used to make an output choice.

The agent gets this information by analyzing the game board. It will use the following Input/Output:

Input : Location of Sheep, Location of fellow Caveman, Own Location
Output : Move to choose (North, South, East, West, Stay)

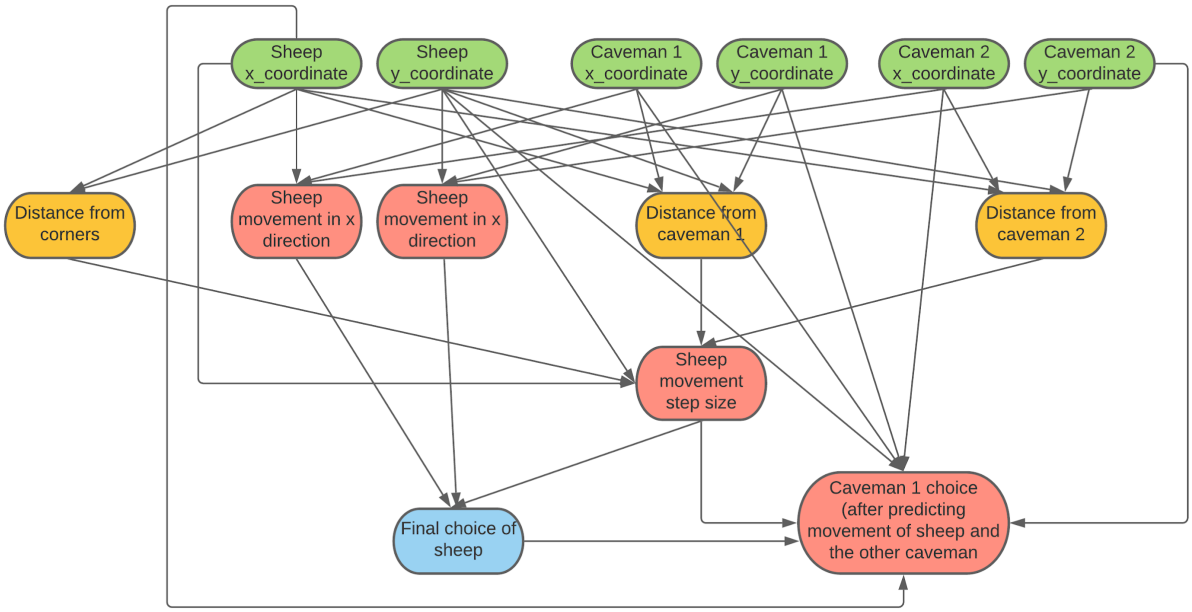


Fig.1.: Combined Caveman and Sheep model. Caveman 1 assumes that caveman 2 uses the min-euclidean distance method to predict the sheep. This means that if we know his location, the sheep's location and our guess of what choice the sheep makes we should be able to learn sheep's strategy from this. The green variables are observable, orange and blue variables are scripted and red variables are to be learned.

Sheep's Choice Decision

Expression for Choice, which combines DirectionX, DirectionY and Step into the final choice of the sheep.

```
if(Step==”Stay”)
        Stay;
if(Step==1)
        if(DirectionX==”East”)
                if(DirectionY==”North”)
                        Move North-East
                else
                        Move South-East
        else
                if(DirectionY==”North”)
                        Move North-West
                else
                        Move South-West
```

```
else
    if(DirectionX==”East”)
        if(DirectionY==”North”)
            Move 2 units North-East
        else
            Move 2 units South-East
    else
        if(DirectionY==”North”)
            Move 2 units North-West
        else
            Move 2 units South-West
```

Training

Run about 100000 iterations of the game using a random start location for each actor in each game until the network is trained sufficiently (the inferences in the bayesian network is stable).

Testing

Play 5 games upto 200 movements and output the agent that wins. See if Caveman 1 wins more.

Tie Braking

In case if more both the cavemen try to move to the same tile, the first cavemen in the list would be allowed to make the move. The order of choice making is the two cavemen, first and second, and then the sheep.

Lookahead

If the cavemen are to utilize their Bayesian Networks, that now are capable of predicting the movement of the other actors in the game, a lookahead is needed. The idea behind the lookahead is that you use the prediction of the other actors for each possible move you may choose. If you find a winning situation within one of these possible moves you choose it, or else you try again and make predictions of your possible moves after your first series of possible moves. This can be done as many times as required.

Note: To keep memory constraints in check, a lookahead of 1 step is used.

We decided to let the caveman make a random move 20% of the time. This would allow situations where the caveman causes the sheep to make a choice it would not have taken.

PART III

Markov Decision Process - MDP (Total = 40 points)

Most of the X-Men are mutants, a subspecies of humans who are born with superhuman abilities activated by the "X-Gene". The X-Men fight for peace and equality between normal humans and mutants in a world where anti-mutant bigotry is fierce and widespread. They are led by Charles Xavier, also known as Professor X, a powerful mutant telepath who can control and read minds. Their archenemy is Magneto, a powerful mutant with the ability to manipulate and control magnetic fields who leads the Brotherhood of Mutants. Both have opposing views and philosophies regarding the relationship between mutants and humans. While the former works towards peace and understanding between mutants and humans, the latter views humans as a threat and believes in taking an aggressive approach against them.

Jean Grey is one of the most beloved X-Men. But when a mission goes wrong, Jean is exposed to a dark and ancient power. This power has destroyed everything it comes in contact with, until her. Now that this power is becoming unstable, she releases it with destruction and anger. Now that this foreign power is consuming her, and the world is threatened, the X-Men have to face an important truth: they must save either the world, or their friend who threatens it. Magneto calls her 'The phoenix' and intends to use her to declare war against humanity.

In this assignment, you will use some algorithms to compute optimal policies in Markov decision processes (MDP's) to help wolverine escape from Magneto while trying to find Jean in order to kill her.

You are given the following grid world where Wolverine and other mutants from Xavier's School for Gifted Youngsters live along with Magneto and his brotherhood of mutants.

5	C2		A		
4					
3		B			
2					C1
1					
	1	2	3	4	5

The wolverine (A) can occupy any of the 24 blank squares. The Magneto (B) also can occupy any square, except for square (5,5) which is Xavier's school of Gifted Youngsters. Jean which can be at C1(5,2) or C2(1,5). Currently, she is at (5,2). Thus, MDP has 24*23*2 = 1104 states.

Wolverine and Magneto can each move one square in any direction - Up,down,left and right **but not diagonal**. They also can choose not to move at all. (4,3) is blocked due to the wall. Thus, there are 5 possible moves from each square. If an action is attempted that causes the characters(Wolverine and Magneto) to bump into a wall, then simply stay at the same location. In this problem, we will always take the point of view of the wolverine.

Reward Policies:

- When Wolverine is at Jean's place, it receives a reward of +20.
- When Magneto is at Wolverine's place, Wolverine receives a reward of -20.
- When the Magneto is at wolverine's place and wolverine is at jean's place, the reward is -15.
- All other configurations have a reward of 0.

Thus, the wolverine is trying to kill Jean while simultaneously avoiding the Magneto.

Jean is always available in exactly one of the two locations listed above. At every time step, Jean remains where she is with 80% probability. With 20% probability, Jean vanishes and reappears at another location.

States are encoded as six tuples, the first two numbers indicating the position of Magneto, the second two numbers the position of Wolverine, and the last two numbers the position of Jean. Thus, 2:3:3:5:5:2 indicates, as depicted in the figure above, that Magneto is in (2,3), Wolverine is in (3,5), and Jean is in (5,2). Magneto and wolverine take alternate moves. However, in encoding the MDP, we collapse both moves into a single state transition. In addition, Jean, when she moves, does so simultaneously with the wolverine's move. For instance, from the configuration above, if the wolverine moves to (2,5) and the Magneto responds by moving to (2,4), while jean moves to (1,5), this all would be encoded as a *single* transition from state 2:3:3:5:5:2 to 2:4:2:5:1:5.

The Wolverine and Magneto have 4 actions available ('UP', 'RIGHT', 'DOWN' and 'LEFT'). Each action moves the Wolverine/Magneto in its direction with probability 0.95. When the wolverine tries to move outside of the grid, the action will have no effect with probability 1. Staying in its own state will happen with probability 0.05.

We will consider two versions of Magneto:

1. In first version, Magneto is dumb and lazy, simply wanders randomly around its environment choosing randomly among its available actions at every step.
2. In the second version, Magneto is intelligent and active. Here, Magneto always heads straight for wolverine following the shortest path possible. Thus, after wolverine makes its move, Magneto chooses the action that will move it as close as possible to the wolverine's new position. (If there is a tie among the Magneto's best available options, the Magneto chooses randomly among these equally good best actions.)

For both versions of Magneto, your job will be to compute the wolverine's optimal policy, i.e. the action that should be taken at each state to maximize the wolverine's expected discounted reward, where we fix the discount factor (gamma) to be 0.85.

Task 1: Implement value iteration for both versions of Magneto on MDP (10 points)

Task 2: Implement policy iteration for both versions of Magneto (10 points).

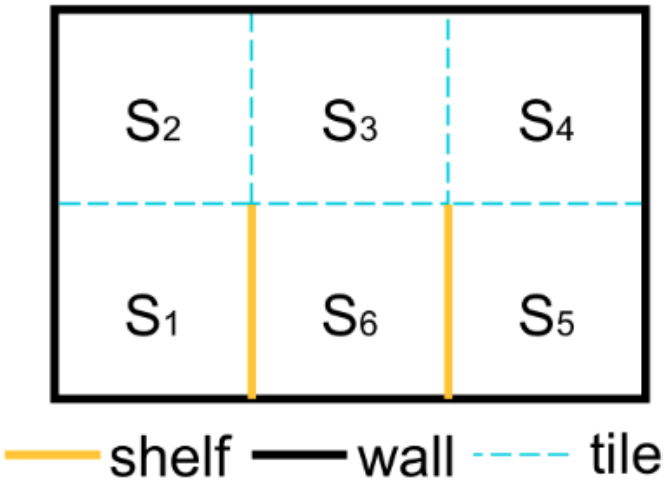
Task 3: Implement and visualize the MDP board and strategy (policy) graphically. (10 points)

Task 4: Compare the results for all the four approaches - Value iteration for lazy Magneto, Value iteration for active Magneto, Policy iteration for lazy Magneto, Policy iteration for active Magneto with proper tables/graphs/statistics. Comment which one is best among all the four mentioned approaches. (10 points)

PART IV
Robot localization using Hidden Markov Model (HMM) [20 Points]

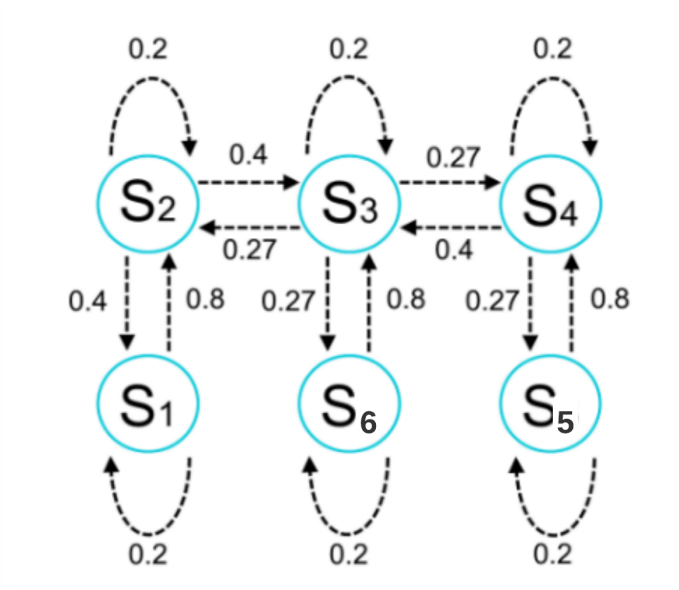
Problem of mobile robot in a warehouse. The agent is randomly placed in an environment and we, its supervisors, cannot observe what happens in the room. The only information we receive are the sensor readings from the robot.

Environment



The agent can move within an area of 6 square tiles. In the mini-warehouse there is one shelf located between tiles S_1 and S_6 and a second shelf between S_6 and S_5

The environment of the agent consists of six discrete states. The time is also discrete. At each subsequent time step the robot is programmed to change its position with a probability of 80% and moves randomly to a different neighboring tile. As soon as the robot makes a move, we receive four readings from the sensing system.



Sensors

Just as we humans can localize ourselves using senses, robots use sensors. Our agent is equipped with the sensing system composed of a compass and a proximity sensor, which detects obstacles in four directions: north, south, east and west. The sensor values are conditionally independent given the position of the robot. Moreover, the device is not perfect, the sensor has an error rate of e = 25%

You need to implement the above as an HMM to track the robot and report its location after every 10 iterations (total 100 iterations) alongwith the estimated probabilities.

Best of Luck!!!