Assignment 2- AI

Submitted By:

Aaditya Raj Barnwal

2020csm1001

Assumptions

Following Assumptions have been made in each and every part:

Part 1: No Assumption Required

Part 2: No Assumption Required

Part 3:

In this part I have assumed that the co-ordinates are starting from 0,0 so that it makes me easy to implement , However I have adjusted the coordinates according so that you cant able to find any difference in the final output , But in the code part I have assumed the coordinates are starting from 0,0 to 4,4.

Part 4:

In this part I have created number of discrepancies at random with an error rate of 25% , if you want to modify or want to give input then all you have to do change a few code and take input the number of discrepancies (in line 95)

①

(i) $\mu = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\Sigma = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}$

The bivariate normal distribution is the statistical .

$$P(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \text{Exp}\left[-\frac{z}{2(1-\rho^2)}\right] \quad \text{—— (i)}$$

where ,

$$z = \frac{(x_1-\mu_1)^2}{\sigma_1} - \frac{2\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2}$$

and $\rho = \frac{V_{12}}{\sigma_1\sigma_2}$, $\sigma_1^2 = \sigma_{11}^2 + \sigma_{12}^2$

$\sigma_2^2 = \sigma_{21}^2 + \sigma_{22}^2$

and $V_{12} = \sigma_{11}\sigma_{21} + \sigma_{12}\sigma_{22}$

Now, the marginal probabilities are :-

$$P(x_1) = \int_{-\infty}^{\infty} P(x_1, x_2)\, dx_2$$

$$= \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \left[\text{Exp} - \left(\frac{z}{2(1-\rho^2)}\right)\right] dx_2$$

$$= \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \int_{-\infty}^{\infty} \text{Exp}\left[\frac{-z}{2(1-\rho^2}\right] dx$$

on solving the above integration

$$P(x_1) = \frac{1}{\sigma_1\sqrt{2\pi}} \operatorname{Exp}\left[-(x_1-\mu_1)^2 / 2\sigma_1^2\right] \qquad \text{—— (ii)}$$

Now for $P(x_2)$

$$P(x_2) = \int_{-\infty}^{\infty} P(x_1, x_2)\, dx_1$$

$$= \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \operatorname{Exp}\left[\frac{-z}{2(1-\rho^2)}\right] dx_1$$

[illegible]

$$= \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \int_{-\infty}^{\infty} \operatorname{Exp}\left[\frac{-z}{2(1-\rho^2)}\right] dx_1$$

where $z = \dfrac{(x_1-\mu_1)^2}{\sigma_1^2} - \dfrac{2\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2} + \dfrac{(x_2-\mu_2)^2}{\sigma_2^2}$

on solving the Integration

$$P(x_2) = \frac{1}{\sigma_2 \sqrt{2\pi}} \operatorname{Exp}\left[-\frac{(x_2-\mu_2)^2}{2\sigma_2^2} \right] \quad \text{——— (iii)}$$

Now using equation (i), (ii) and (iii), we can get

$$P(x_1/x_2) = \frac{P(x_1, x_2)}{P(x_2)}$$

$$= \frac{\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \operatorname{Exp}\left[-\frac{1}{2(1-\rho^2)} \left[\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - 2\frac{\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_x\sigma_y} \right] \right]}{\frac{1}{\sigma_2\sqrt{2\pi}} \operatorname{Exp}\left[-\frac{(x_2-\mu_2)^2}{2\sigma_2^2} \right]}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_1\sqrt{1-\rho^2}} \operatorname{Exp}\left[-\frac{1}{2\sigma_1^2(1-\rho^2)} \left[x_1-\mu_1 - \rho\frac{\sigma_1}{\sigma_2}(x_2-\mu_2) \right]^2 \right\}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_1\sqrt{1-\rho^2}} \operatorname{Exp}\left[-\frac{1}{2}\left[\left(\frac{x_1-a}{b} \right)^2 + c \right] + \frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2} \right)^2 \right.$$

where $a = \mu_1 + \rho\frac{\sigma_1}{\sigma_2}(x_2-\mu_2)$, $b = \sigma_1\sqrt{1-\rho^2}$

and $c = \left(\frac{x_2-\mu_2}{\sigma_2} \right)^2$

Therefore,

$$P(x_1/x_2) = \frac{1}{\sqrt{2\pi}\ \sigma_1\sqrt{1-\rho^2}} \ \mathrm{Exp}\left\{ -\frac{1}{2\sigma_1(1-\rho^2)}\left[x_1-\mu_1-\rho\frac{\sigma_1}{\sigma_2}(x_2-\mu_2)\right]^2 \right\}$$

where $\rho = \frac{V_{12}}{\sigma_1\sigma_2}$ where, $V_{12} = \sigma_{11}\sigma_{21} + \sigma_{12}\sigma_{22}$

and $\sigma_1^2 = \sigma_{11}^2 + \sigma_{12}^2$

$\sigma_2^2 = \sigma_{22}^2 + \sigma_{21}^2$

~~Similarly~~ Similarly

$$P(x_2/x_1) = \frac{1}{\sqrt{2\pi}\ \sigma_2\sqrt{1-\rho^2}} \ \mathrm{Exp}\left\{ -\frac{1}{2\sigma_2^2(1-\rho^2)}\left[x_2-\mu_2-\rho\frac{\sigma_2}{\sigma_1}(x_1-\mu_1)\right]^2 \right\}$$

Problem 1.B- Please refer to the code

Problem 1.C-

When a=0

Estimated value for Mean, Mean of x1 = 1.0054880982780774

Mean for x2 = 1.9870341218512184

Estimated value of Sigma= [[1.02006207 0.00453068]

[0.00453068 1.01433323]]

After Burning for 2000 (Discarding first 2000 values)

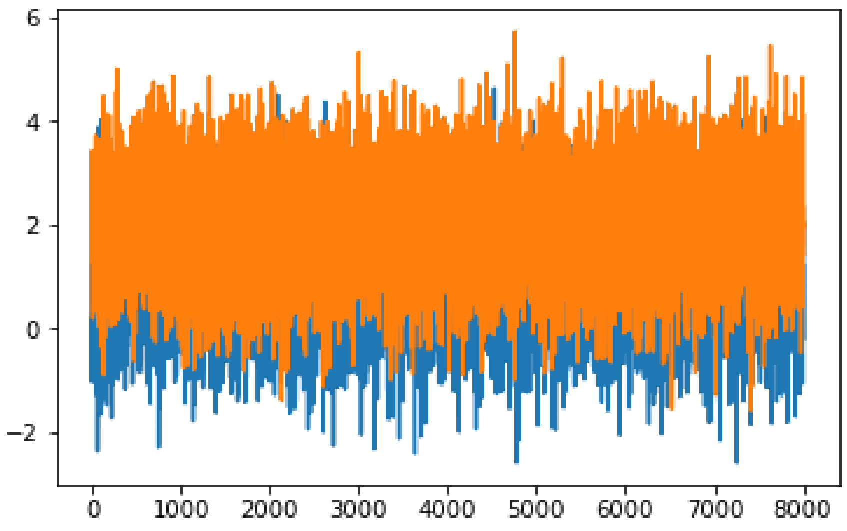
Comment:-

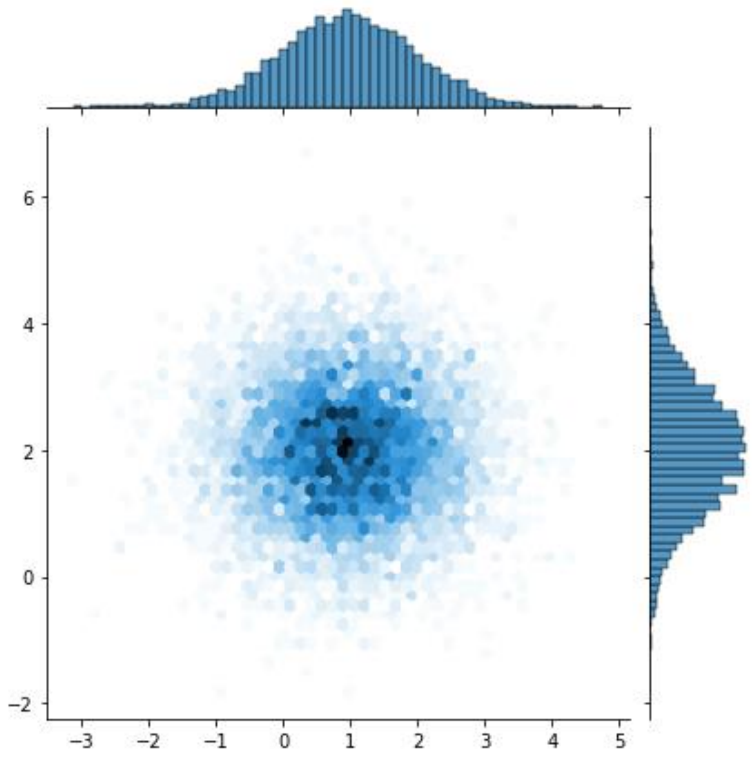
After discarding some points (2000) the plot converges, and It shows the true distribution of x1 and x2

The mean deviation of x1 is 0.005439

The mean distribution for x2 is 0.023809

The sigma variation is around 0.02006207





Problem 1.D –

When a=0.99

Estimated value for Mean, Mean of x1 = 1.0038398719524328

Mean for x2 = 2.0038095080011815

Estimated value of Sigma= [[1.0156995 1.01586552]

[1.01586552 1.01612904]]

After Burning for 2000 (Discarding first 2000 values)

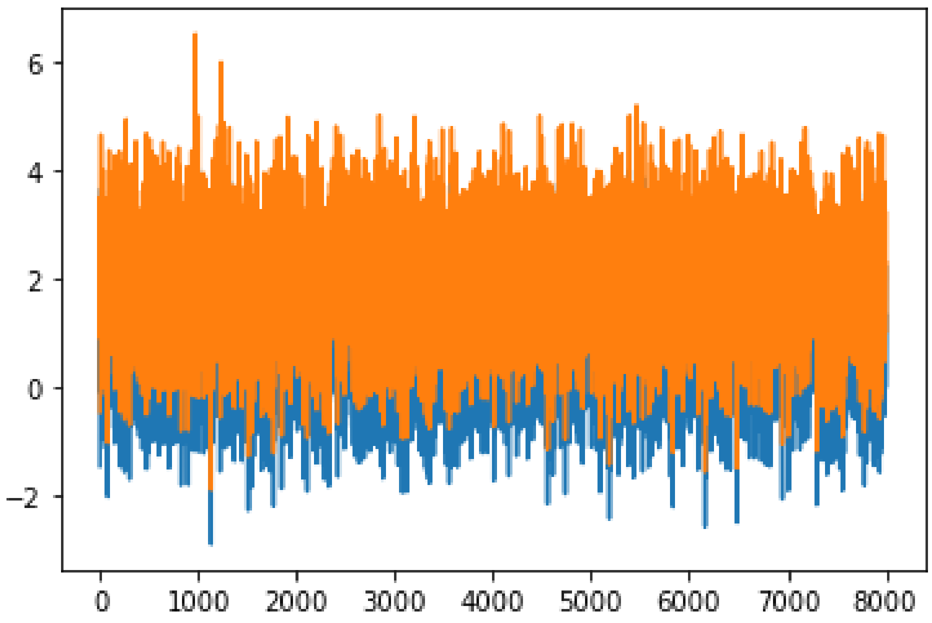
Comment:-

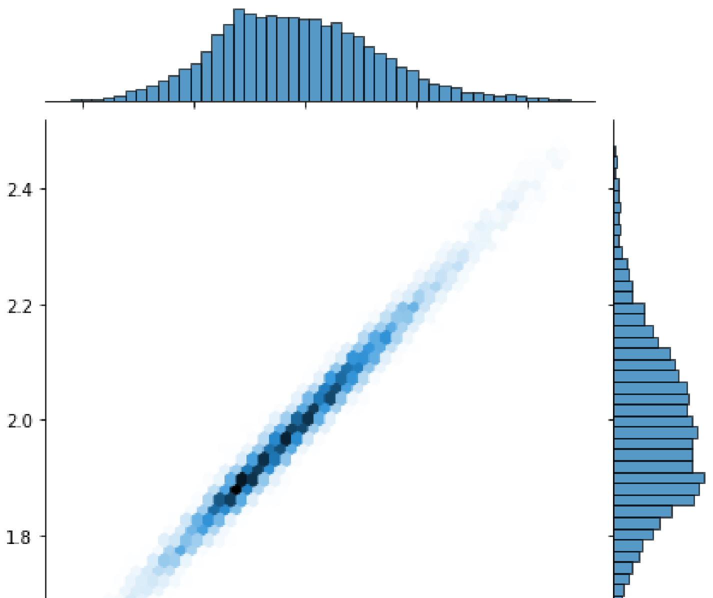
After discarding some points (2000) the plot converges, and It shows the true distribution of x1 and x2

The mean deviation of x1 is 0.003839

The mean distribution for x2 is 0.003809

The sigma variation is around 0.01569





Part 2:

Prey and Predator model using Bayesian Network

Without Using Bayesian Network (200 iteration steps)	With Using Bayesian Network (200 iteration steps)
1. Caveman2 wins the game and catches the Sheep	1. Caveman1 wins the game and catches the sheep
2. No one Catches the sheep and therefore sheep escaped	2. Caveman1 wins the game and Catches the sheep
3. Caveman1 wins the game and catches the sheep	3. Caveman1 wins the game and catches the sheep
4. Caveman2 wins the game and catches the sheep	4. Caveman1 wins the game and catches the sheep
5. Caveman1 wins the game and catches the sheep	5. Caveman2 wins the game and catches the sheep

Analysis:

In case of without Bayesian Network (BN), Sometimes sheep is able to escape and sometimes it got captured by the cavemen,

As everyone is using the distance formula (Euclidean distance) and travel accordingly, Cavemen move in such a way that it minimizes its distance from sheep and sheep uses it so that it can stay as far as from Cavemen and should not be trapped in any corner.

And as everyone is starting with random initial position, and it may be sometimes possible that a caveman (1 or 2) may got a position just adjacent to the sheep and it catches the sheep and game ends

In case of using BN, the caveman1 is always using BN to predict the next move of the sheep , as it is trained in 10K iteration and it uses the CPT created in those iteration to predict the next location of the sheep and hence it catches the sheep maximum number of times , also it uses random position with a probability of 20% ,whereas the sheep and other caveman uses simple distance formula to fetch the next move of their prey and predator ..

Here also, sometimes may be possible that caveman2 will win the game , this is because as it is using random location , so caveman2 may land adjacent to sheep and catches the sheep and game ends , But this event is very unlikely

So , The Conclusion is Caveman1 will win the game ,when its using the BN

PART III:
Markov Decision Process - MDP

Task 1: Please See the code of this Task, I have created two files for Value Iteration, first files contains the code of Value Iteration when Magneto is Lazy, he simply wanders randomly around its environment and jean switches its position (between c1 and c2) with a probability of 80%. One the other hand second file is for Value Iteration when magneto is smart and it move in such a way that it minimize the distance between wolverine and magneto

Task 2: Please find the code of this Task, Here also I have created 2 files for policy iteration, First file contains the code when magneto is lazy, and he makes random move and in the second file , magneto is smart and it always head towards wolverine, same as Task 1, but here we use Policy Iteration

Discount factor=0.85

Task 3:

Here we visualize it **Value Iteration**

```
| jean | Left  | Left  | Left  | Left   |
| Up   | Left  | Left  | Left  | Down |
| Up   | Left  | Left  |WALL| Down|
| Up   | Up    | Right | Up    | Down  |
| Up   | Left  | B     | Right | Right |
```

This is for Value_iteration_Lazy Magneto

In the above diagram you can clearly see the final optimal policy that the wolverine should take,

Diagram Explanation: Here jean in the above diagram indicates that at that time jean is standing at c2 location and Magneto(with B representation) is standing in the left corner as magneto is moving randomly in the environment, Here WALL represent that at that location Wall is present so that no one can move there (So I am displaying it as WALL), Wolverine is represented by 'A' (as per question), Here left, right, up, down are the directions in which one should take to get the maximum rewards to reach the goal(Jean position), Here you cant see the wolverine its because it reaches the Jean Position and reaches the Goal state

Value Iteration (Lazy Magneto) : Here you can clearly see the above diagram how wolverine should move such a way that it reaches the Goal state(Jean Position) with maximum rewards

```
| B    | Left | Left | Up   | Left |
| Up   | Left | Left | Left | Down |
| Up   | Left | Left | WALL | Down |
| Up   | Up   | Right| Up   | Down |
| Up   | Down | Down | Up   | Right|
```

This is for smart magneto value iteration

In the above diagram you can clearly see the steps that wolverine should take to reach the Goal state(Jean Position)

Diagram Explanation: You can see that there is no jean in the above diagram only you see is that Magneto , This is because as Magneto is Smart now, he head towards the wolverine and wolverine is following Jean and Jean is overlapped by wolverine and wolverine is overlapped by magneto ,so can't see them at c2 location.

Value Iteration (Smart Magneto) : You Can clearly see the difference between the previous result and current result, In previous case Magneto is lazy and makes random moves so he is unable to catch the wolverine and in the second case he is able to capture the wolverine.

Now we will see the result of **Policy Iteration:**

```
| jean | Left | Left | Right | Down |
| Up   | Up   | Left | Right | Down |
| Up   | Left | Right| WALL  | Down |
| Up   | B    | Right| Right | Up   |
| Left | Down | Right| Right | Up   |
```

This is for Policy Iteration when Magneto makes random moves

In the above diagram you can clearly see the final optimal policy that the wolverine should take,to reach the goal (Jean Position),when Magneto is making random moves

Diagram Explanation: You Can see the result matrix , what movement wolverine should take to reach to Jean Position , as Magneto is travelling randomly , so its far away from wolverine and wolverine and jean and wolverine are at position at c2 and its not showing because at that location character 'A' is overlapped by jean that's why its not showing

Policy Iteration (Lazy Magneto): Here you can clearly see the above diagram how wolverine should move such a way that it reaches the Goal state(Jean Position) with maximum rewards

B	Left	Left	Right	Down
Up	Up	Down	Right	Down
Up	Right	Down	WALL	Down
Right	Right	Right	Right	Up
Right	Right	Right	Right	Up

This is for Policy Iteration (Smart Magneto)

In the above diagram you can clearly see the steps that wolverine should take to reach the Goal state(Jean Position

Diagram Explanation: You can see the movement at each step wolverine should take to reach the goal state(Jean Position) , You can compare this result with the Value iteration(Smart Magneto) result, Here also, the same case, you cant see the position of Jean and wolverine because it being overlapped by Magneto , As Magento is Smart and it head towards the wolverine and wolverine is for Jean , If you see the compare the previous result of value iteration with policy iteration when magneto is Smart, you will find that only some movement in some steps has been changed , this is because Jean is switching her location with a probability of 80% , so the path may change if you run the same program again and again

Task 4:

The 4 results obtained from Value Iteration and Policy iteration when Magneto is Lazy and When magneto is Smart are:

First I will compare the VI when Magneto is Lazy

jean	Left	Left	Left	Left
Up	Left	Left	Left	Down
Up	Left	Left	WALL	Down
Up	Up	Right	Up	Down
Up	Left	B	Right	Right

PI , when Magneto is Lazy

jean	Left	Left	Right	Down
Up	Up	Left	Right	Down
Up	Left	Right	WALL	Down
Up	B	Right	Right	Up
Left	Down	Right	Right	Up

Analysis:

From the above result, we can clearly see, The wolverine always reaches the goal by ,Almost completely avoiding the magneto and magneto is also not able to block the wolverine as magneto is taking random movement,

From the above diagram, we see that only jean and magneto are visible not wolverine , this is because (as explained earlier) wolverine position is overlapped by the jean so wolverine is not visible but its there only at the jean position .

While getting this output what I analyze that value iteration takes more number of steps as compare to policy iteration

Now See the VI and PI of magneto using smart move towards wolverine

```
| B    | Left  | Left  | Up    | Left  |
| Up   | Left  | Left  | Left  | Down  |
| Up   | Left  | Left  | WALL  | Down  |
| Up   | Up    | Right | Up    | Down  |
| Up   | Down  | Down  | Up    | Right |
```

```
| B    | Left  | Left  | Right | Down |
| Up   | Up    | Down  | Right | Down |
| Up   | Right | Down  | WALL  | Down |
| Right| Right | Right| Right | Up    |
| Right| Right | Right| Right | Up   |
```

Analysis:

Here Also , you can see the result and compare it with one another, In this case you only see is magneto, This is because , as explained earlier , the jean and wolverine and magneto are at the same position and so the value is got overlapped and we only see the magneto, you can also verify the same in the coding part and print the position of jean and wolverine(if necessary)

Value Iteration: Value iteration computes the optimal state value function by iteratively improving the estimate of **V(s)**. The algorithm initialize **V(s)** to arbitrary random values. It repeatedly updates the **Q(s, a)** and **V(s)** values until they converges. Value iteration is guaranteed to converge to the optimal values.

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

Value iteration: It computes the optimal state value function by iteratively improving the estimate of **V(s)**. The algorithm initializes **V(s)** to arbitrary random values. It repeatedly updates the **Q(s, a)** and **V(s)** values until they converges. Value iteration is guaranteed to converge to the optimal values.Its a 2 step process,

The first is called **Policy Extraction**, which is how you go from a value to a policy — by taking the policy that maximizes over expected values.

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^{\pi_i}(s')\right]$$

The second step is **Policy Evaluation.** Policy evaluation takes a policy and runs value iteration *conditioned on a policy*. The samples are forever tied to the policy, but we know *we have to run the iterative algorithms for way fewer steps to extract the relevant **action** information*

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') \left[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')\right]$$

For PI, we perform policy evaluation for some policy, meaning that we wait until convergence to the value function of the current policy. Nevertheless, instead of waiting until convergence, we could tolerate some error, truncate the policy evaluation process at some point and perform the policy improvement step over the approximate value function. Now, consider the extreme case where we only apply one iteration of the policy evaluation loop and then improve the policy. This extreme case is equivalent to applying the optimal Bellman operator. In other words, we can see VI as an extreme case of PI, where the policy evaluation step is approximated with just one single iteration.

Regarding complexity, note that policy evaluation implies solving the Bellman equation, which could be accomplished by, e.g., solving the linear system of equations (using numerical linear algebra) or by finding its fixed point (iterating the Bellman operator). Typically, for large state-action sets, the complexity of finding the fixed point is much smaller than the complexity of solving the system of equations. But, in any case, policy evaluation is costly. We can compare the complexity of VI and PI:

- Double loop vs. single loop:
    - Policy iteration requires (surprisingly) few iterations of the outer loop, although each of these iterations requires a whole inner (policy evaluation) loop inside. Indeed, for finite-state MDP's, there exists a finite number of deterministic policies. Hence, PI must converge to the optimal policy in a finite number of steps.
    - Value iteration requires several iteration of its single loop, but each iteration requires no further loop.
- Maximization over the action set, to consider if the action set is huge and maximization is very costly:
    - Policy iteration performs the maximization over the action set at each iteration of the outer loop (which, again, it requires very few iterations).
    - Value iteration performs the maximization over the action set at each iteration of the inner loop (which usually requires many iterations to converge).

Part 4: Robot localization using Hidden Markov Model

Analysis:

In this Problem, we have to predict the location of the robot , which is moving inside the grid using HMM,

The error rate is 25%, and with 80% probability it changes its direction randomly and report its location in every 10th step (out of 100 iteration).

I am using Filtering to estimate the location of robot,

The initial probability of all the states is 0.1667 (1/6) and Transition probability is given the question,

given that we are in state *i*, what is the probability that the sensor returns reading E=j

$O_{i,j} = P(E=j / X=i) = (1-e)^{4-d} e^4$, where e is error (0.25) and d is the discrepancy- a number of signals that are different- between the true values for tile *i* and the actual reading.

In this problem, I am running the loop 100 times and first I am getting the sensor values and then again I am with an error of 25% generating the discrepancy and then I am printing the state and probability of that state

The Following Results are obtained the following are the state and probability pair
State is 4 With a Probability of 0.6208810572687224
State is 3 With a Probability of 0.5507038345489472
State is 2 With a Probability of 0.6623432737938458
State is 4 With a Probability of 0.4006772489074379
State is 2 With a Probability of 0.7210637942343241
State is 4 With a Probability of 0.542413360651528
State is 2 With a Probability of 0.4297473016227128
State is 5 With a Probability of 0.8195062727640631
State is 3 With a Probability of 0.4984867759805794
State is 2 With a Probability of 0.8324843626140205

Analysis:

The probability after every 10 step is as shown in the above table: here one thing is of note that the robot is going to a random location with a probability of 20% , And also I am printing that state only whose probability is maximum in that state after observing those sensor values