Reinforcement Learning

Lab Assignment 3

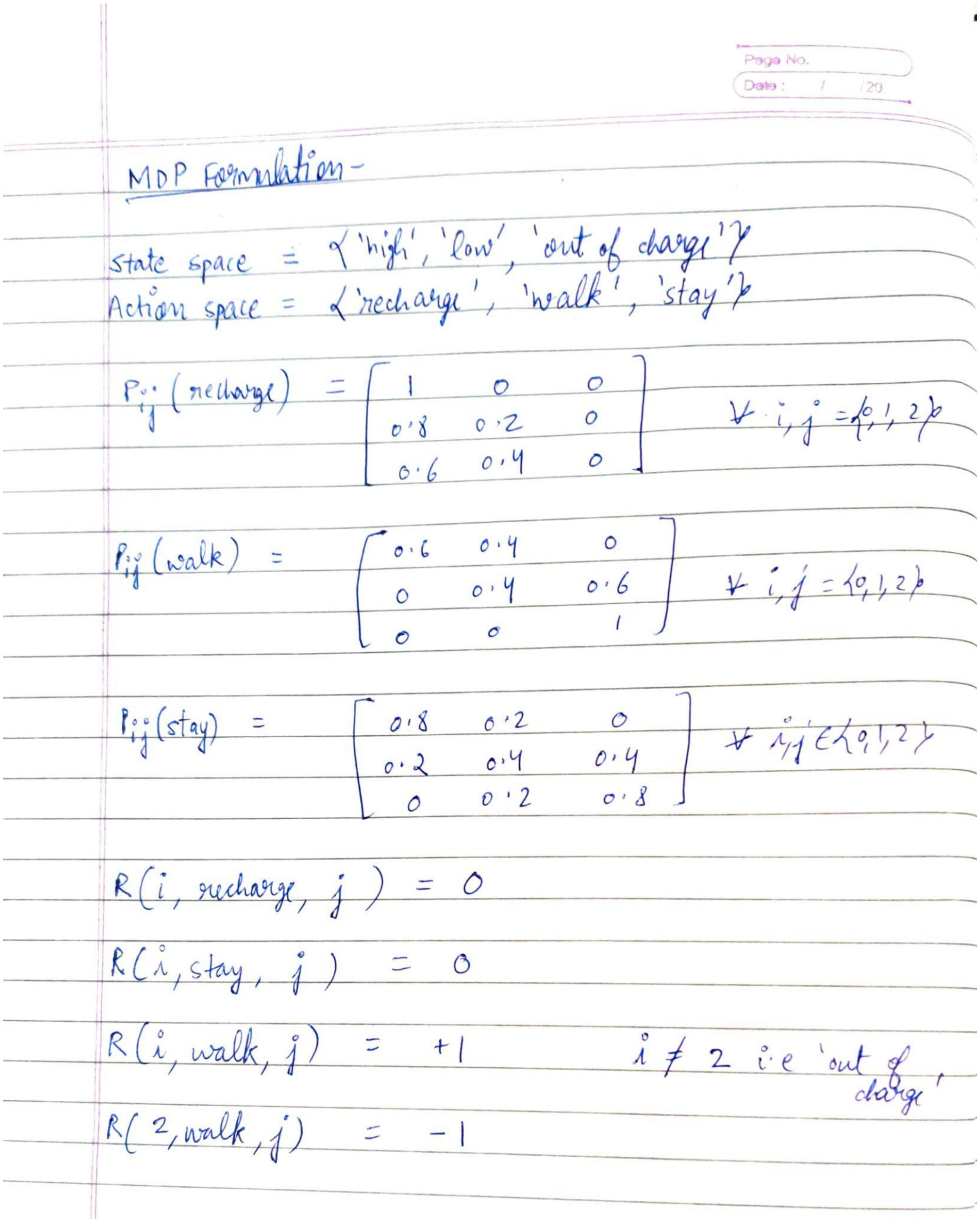
190020007-190020021-190020039

MDP 1: 190020021

Problem

There is an automatic garbage collecting robot that has been implemented in a city.Its battery percentage is divided into three types: High(50-100%), Low(0-50%) and out of charge. Also, to save battery it has three modes: Walk,Stay and Recharge. The robot cannot be used for collecting garbage if Recharge mode is on. During walk mode a reward of +1 is awarded for each piece of garbage collected.Also, there is a stay mode which is basically energy saving in which robot stands at a place and people have to throw materials inside it rather than it picking them.No reward is given during stay mode as people have to help to clean their surroundings. Furthermore, each time step robot decides to walk its battery might drain and it may end up in the Low state. Also, if the robot decides to stay then too it consumes charge but at a lower rate so there are less chances of running out of battery. Once the robot reaches ‘out of charge’ state it can only recharge and do nothing else.

MDP Formulation:



Page No.

Date : / /20

MDP Formulation -

State space = {'high', 'low', 'out of charge'}

Action space = {'recharge', 'walk', 'stay'}

$$P_{ij}(\text{recharge}) = \begin{bmatrix} 1 & 0 & 0 \\ 0.8 & 0.2 & 0 \\ 0.6 & 0.4 & 0 \end{bmatrix} \quad \forall\, i, j = \{0, 1, 2\}$$

$$P_{ij}(\text{walk}) = \begin{bmatrix} 0.6 & 0.4 & 0 \\ 0 & 0.4 & 0.6 \\ 0 & 0 & 1 \end{bmatrix} \quad \forall\, i, j = \{0, 1, 2\}$$

$$P_{ij}(\text{stay}) = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.2 & 0.4 & 0.4 \\ 0 & 0.2 & 0.8 \end{bmatrix} \quad \forall\, i, j \in \{0, 1, 2\}$$

$R(i, \text{recharge}, j) = 0$

$R(i, \text{stay}, j) = 0$

$R(i, \text{walk}, j) = +1$ $\quad i \neq 2$ i.e 'out of charge'

$R(2, \text{walk}, j) = -1$

Observations while coding:

- **Number of Iterations to converge**

  Policy Iteration: 3

  Value Iteration: 332

  As expected policy iteration converges faster than value iteration for considered convergence factor (eps = 0.01).
- **Different convergence factors**

  The smaller we make eps (convergence factor) the more iterations value iteration took to converge whereas policy iteration took the same number of iterations to converge.
- **Time taken by algorithms**

  Time taken was roughly the same for both of the algorithms for my case as the number of states and actions are small.
- **Optimal Stationary Deterministic Policy Obtained**

  ```
  {'high': 'walk', 'low': 'recharge', 'out_of_charge': 'recharge'}
  ```

Ways to check if optimal policy obtained:

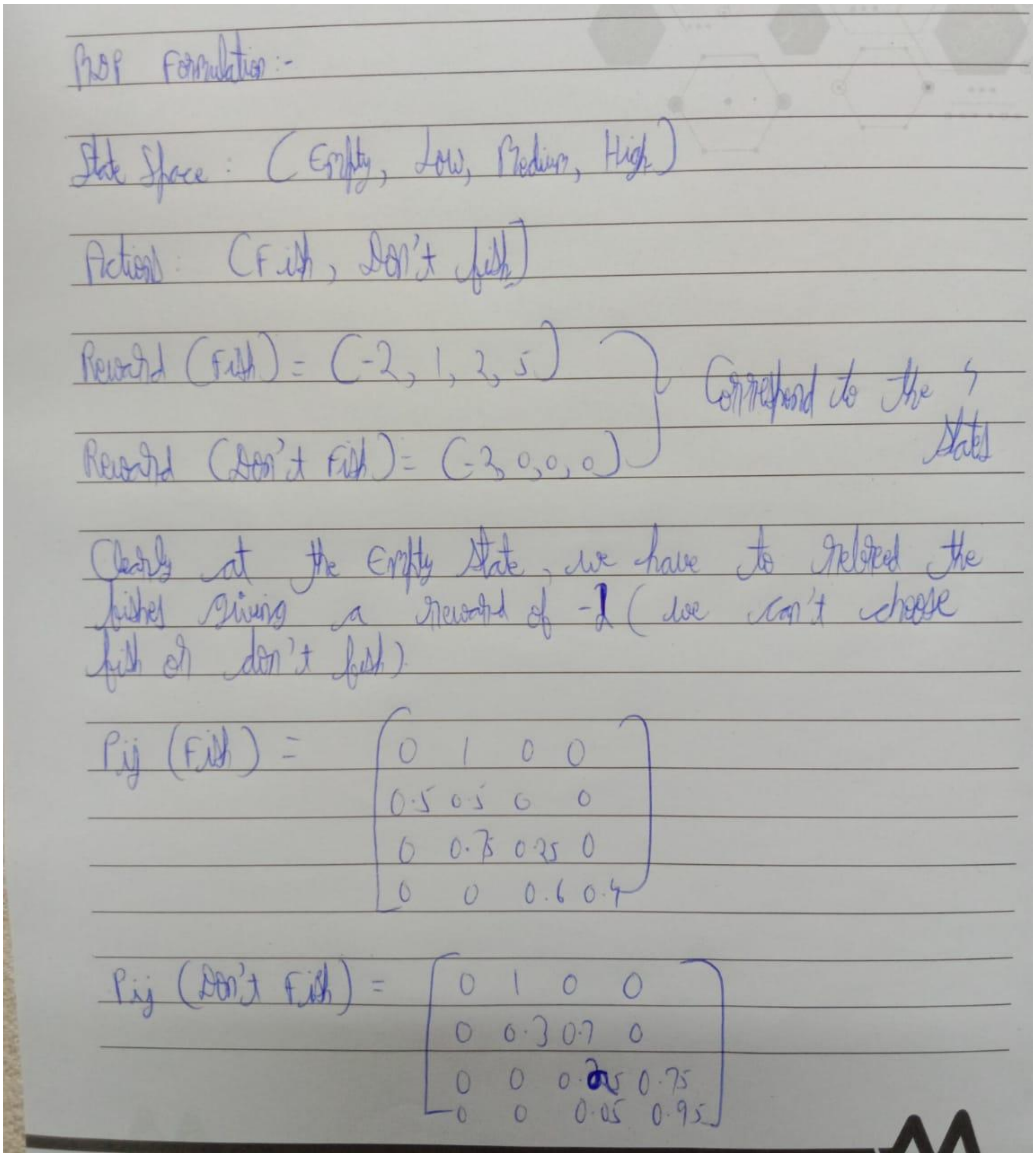
- One naive way to check is to form all stationary deterministic policies and then simulate the known environment for a large number of steps (maybe 1000) and average result over 100 runs.The average rewards obtained by each policy can be compared which can provide us with the best policy. But this becomes computationally expensive as the number of states and actions increase.For my case alone the number of stationary deterministic policies will be 27.
- Another war is to check if the value function obtained is a fixed point of corresponding operators or not. This will help us identify if we have reached optimal value functions or not. Though obtained optimal policy may not be the only one as there can be multiple policies which have same value functions.

MDP2 : 190020007

Fishing:-

The setting of my problem is as follows :-

There is a river which has some fishes. The river can be in any of the 4 states: Empty, Low, Medium, High. These states correspond to the fishes in the river. Two actions are possible, Fish and Don’t Fish. The problem formulation nis as follows :-


MDP Formulation :-

State Space : (Empty, Low, Medium, High)

Actions : (Fish, Don't fish)

Reward (Fish) = (-2, 1, 3, 5)

Reward (Don't Fish) = (-3, 0, 0, 0)

} Correspond to the 4 States

Clearly at the Empty State, we have to rebreed the fishes giving a reward of -2 (we can't choose fish or don't fish)

$$P_{ij}\,(\text{Fish}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.75 & 0.25 & 0 \\ 0 & 0 & 0.6 & 0.4 \end{bmatrix}$$

$$P_{ij}\,(\text{Don't Fish}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.3 & 0.7 & 0 \\ 0 & 0 & 0.25 & 0.75 \\ 0 & 0 & 0.05 & 0.95 \end{bmatrix}$$

Optimal policy for the 4 states : [rebreed, fish, fish, fish] (given by both value and policy iteration).

Policy iteration: I have started with the initial policy don't fish. I have calculated the Vμ for this policy. Then I have calculated the max of Qμ(i, a) over my 2 a's fish and don't fish and have accordingly updated μ. I have repeated these steps until convergence of policy μ.

Observations:

No of iterations required 1: The policy directly converges to optimal policy [rebreed, fish, fish, fish] in 1 step.

Value iteration : In Value iteration, I have simply calculated Q(i, a) for both my a's by starting my original Q(i, a) as 0 and iterating over $Q(i, a) = reward[i] + \Sigma P_{ij} * Q(i, a)$. Then my optimal value function and policy can be found by taking $max_a Q(i, a)$.

Observations : NoI of iterations taken = 22 to converge to final policy [rebreed, fish, fish, fish].

MDP 3 : 190020039

Problem: A simple dairy cow replacement problem (problem taken from “Herd Management Science” by Kristensen, Toft).

For any dairy cow it is relevant to consider at regular time intervals whether it should be kept for an additional period or it should be replaced by a heifer. At the beginning of each lactation period, we observe the state of the animal in production. In this very simple example we assume that the only relevant information is whether the cow is low, average or high yielding. Thus we have got one state variable (milk yield) and three states. Having observed the state, we have to take action concerning the cow. We assume that the action is either to keep the cow for at least an additional period or to replace it by a heifer at the end of the period. If the action replace is taken, we assume that the replacement takes place at the end of the period at some cost. If a cow has been low yielding during a stage, there is a large risk that it will also be low yielding during the following stage if it is kept. On the other hand, if it is replaced, we assume that there are equal probabilities of the new heifer to be low, average or high yielding. All transition probabilities are shown in the problem formulation section. Our problem is now to determine an optimal policy, which in some sense maximizes the net returns of the dairy farmer in an infinite horizon setting.

MDP formulation:

State space, S = {0:"low", 1:"average", 2:"high"}

Action space, A = {0:"keep", 1:"replace"}

Transition probabilities:

Transition probabilities from stage i to stage j following action a						
Present state i	a = 0 (keep)			a = 1 (replace)		
	j = 0 (L)	j = 1 (A)	j = 2 (H)	j = 0 (L)	j = 1 (A)	j = 2 (H)
i = 0 (L)	0.6	0.3	0.1	1/3	1/3	1/3
i = 1 (A)	0.2	0.6	0.2	1/3	1/3	1/3
i = 2 (H)	0.1	0.3	0.6	1/3	1/3	1/3

Expected Immediate Rewards:

Expected immediate reward depending on stage i and following action a		
state i	a = 0 (keep)	a = 1 (replace)
i = 0 (L)	10	9
i = 1 (A)	12	11
i = 2 (H)	14	13

The cost for replacement is taken to be -1.

Discount factor, Gamma = 0.85

Observations from code:

- Stopping criteria: error tolerance, eps = 1e-6

  For value iteration: |V_new(i) - V(i)| < eps, for all i

  For policy iteration: mu_new(i) = mu(i), for all i

  For policy evaluation: |V_mu_new(i)-V_mu(i)| < eps, for all i

- Optimal Stationary Deterministic Policy Obtained: for gamma = 0.85
  [1 0 0] i.e. ['replace' 'keep' 'keep']
  Changing the value of gamma may change the optimal policy.
  For gamma closer to 0.5, optimal policy is [0 0 0] i.e. ['keep' 'keep' 'keep']

- Number of Iterations to converge

  Policy Iteration: 2

  Value Iteration: 102

  The Number of Iterations to converge for value iteration depends on gamma and eps value.

  The Number of Iterations to converge for policy iteration depends on gamma only.

- Time taken to converge

  Policy Iteration: 0.005787699999999951 s

  Value Iteration: 0.0009075999999998974 s

The time taken to converge for policy and value iteration depends on gamma and eps value.

Issues faced:

If the cost for replacement is taken to be less than or equal to -2, the policy iteration and value iteration would give different optimal policies.

Improvements to code to fix above issue:

In the initial code, while calculating Q from R,P,V_mu in policy evaluation, Q was not initialized beforehand and there were some issues with Q not being assigned with the values as desired.
On initializing Q to a matrix of zeros beforehand, the issue was resolved.
Both policy and value iteration give the same optimal policy now.