Assignment 2

2018114016-Saransh Rajput,2018101008-Shivam Mangale

INTRODUCTION

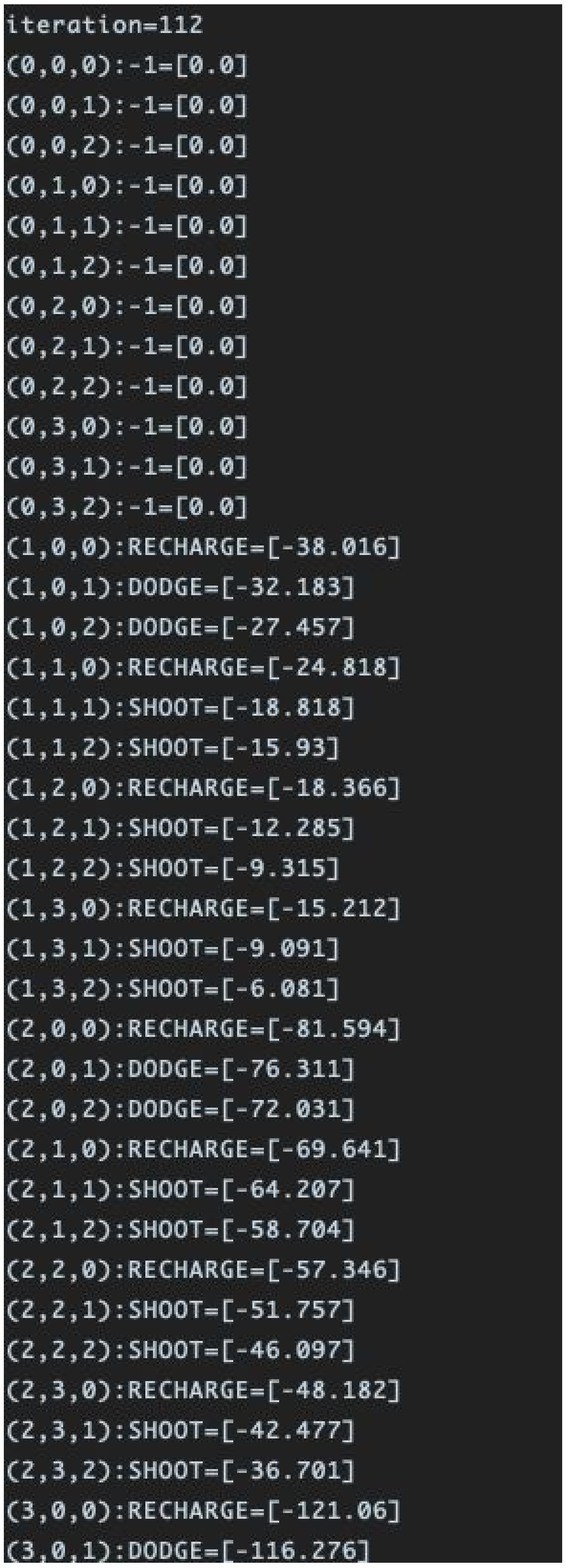
This assignment aims at implementing and understanding the value iteration in machine learning by imagining the story of a hero attacking an enemy dragon. We make use of the following libraries in order to do this project:
os(for mkdir)

Task 1

We run the value iteration algorithm on a 3D state space with the hero's stamina, arrows stored and dragon's health. We give the reward of killing the dragon(health=0) as +10 and appropriate stepcosts. We find that it converges after a limited iterations. We are able to print the appropriate policies at each state in each iteration.

This is done using Value Iteration Algorithm.

It converges at iteration number 112.
In the last iteration it can be observed that whenever the hero is in a state with nonzero stamina and nonzero arrow count he will always shoot. If he does not have stamina he will recharge to get stamina. If he does not have arrows but has stamina he will dodge to get an arrow.
This follows in iterations.

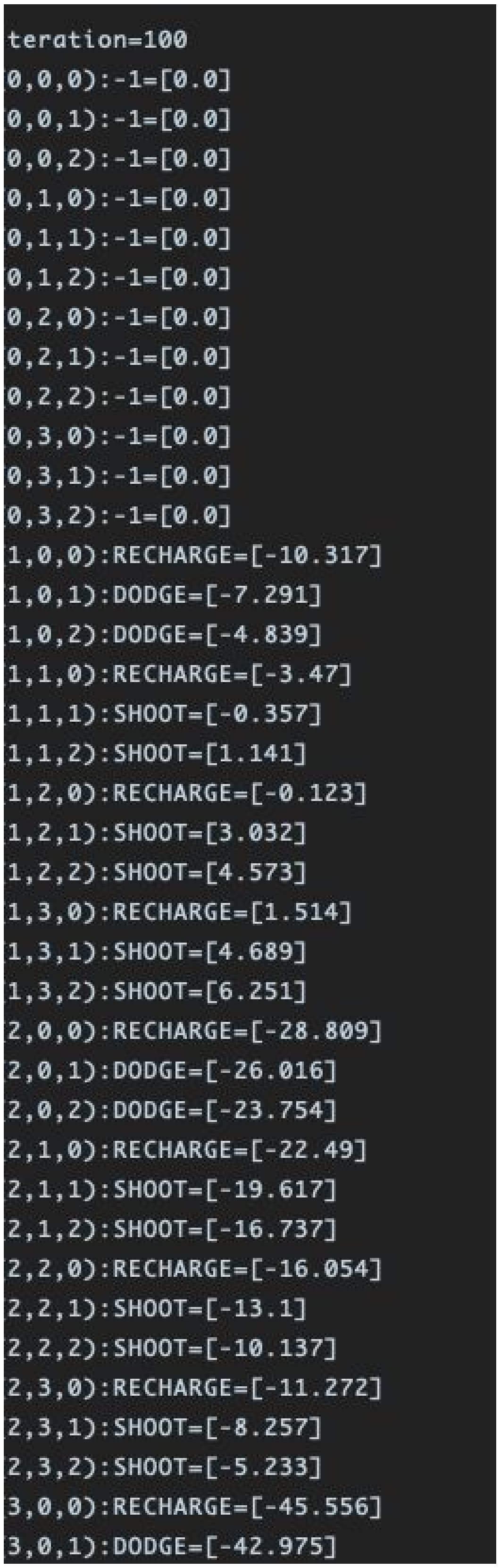

```
iteration=112
(0,0,0):-1=[0.0]
(0,0,1):-1=[0.0]
(0,0,2):-1=[0.0]
(0,1,0):-1=[0.0]
(0,1,1):-1=[0.0]
(0,1,2):-1=[0.0]
(0,2,0):-1=[0.0]
(0,2,1):-1=[0.0]
(0,2,2):-1=[0.0]
(0,3,0):-1=[0.0]
(0,3,1):-1=[0.0]
(0,3,2):-1=[0.0]
(1,0,0):RECHARGE=[-38.016]
(1,0,1):DODGE=[-32.183]
(1,0,2):DODGE=[-27.457]
(1,1,0):RECHARGE=[-24.818]
(1,1,1):SHOOT=[-18.818]
(1,1,2):SHOOT=[-15.93]
(1,2,0):RECHARGE=[-18.366]
(1,2,1):SHOOT=[-12.285]
(1,2,2):SHOOT=[-9.315]
(1,3,0):RECHARGE=[-15.212]
(1,3,1):SHOOT=[-9.091]
(1,3,2):SHOOT=[-6.081]
(2,0,0):RECHARGE=[-81.594]
(2,0,1):DODGE=[-76.311]
(2,0,2):DODGE=[-72.031]
(2,1,0):RECHARGE=[-69.641]
(2,1,1):SHOOT=[-64.207]
(2,1,2):SHOOT=[-58.704]
(2,2,0):RECHARGE=[-57.346]
(2,2,1):SHOOT=[-51.757]
(2,2,2):SHOOT=[-46.097]
(2,3,0):RECHARGE=[-48.182]
(2,3,1):SHOOT=[-42.477]
(2,3,2):SHOOT=[-36.701]
(3,0,0):RECHARGE=[-121.06]
(3,0,1):DODGE=[-116.276]
```

Task 2

We appropriately run the codes for each of the parts with appropriate values for each constant and observe the outputs.

Part 1

We find that the hero heavily prefers shooting over other actions as the step cost is lesser for shooting as compared to others. Also the hero will end up achieving the reward more often as compared to the task 1.

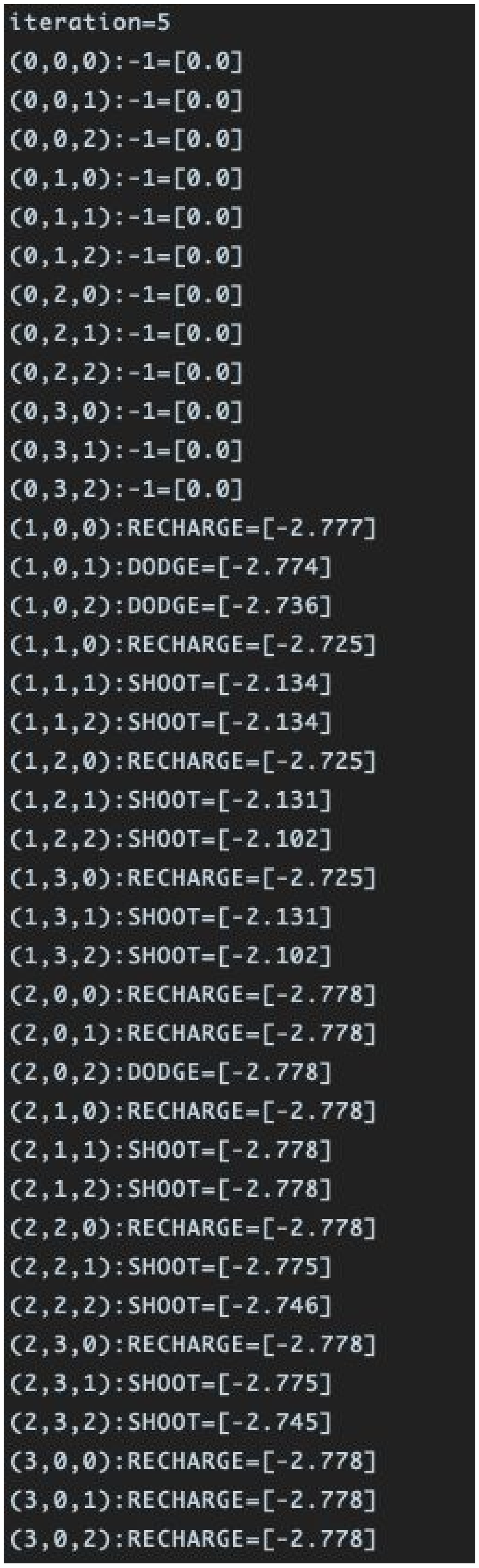


```
teration=100
0,0,0):-1=[0.0]
0,0,1):-1=[0.0]
0,0,2):-1=[0.0]
0,1,0):-1=[0.0]
0,1,1):-1=[0.0]
0,1,2):-1=[0.0]
0,2,0):-1=[0.0]
0,2,1):-1=[0.0]
0,2,2):-1=[0.0]
0,3,0):-1=[0.0]
0,3,1):-1=[0.0]
0,3,2):-1=[0.0]
1,0,0):RECHARGE=[-10.317]
1,0,1):DODGE=[-7.291]
1,0,2):DODGE=[-4.839]
1,1,0):RECHARGE=[-3.47]
1,1,1):SHOOT=[-0.357]
1,1,2):SHOOT=[1.141]
1,2,0):RECHARGE=[-0.123]
1,2,1):SHOOT=[3.032]
1,2,2):SHOOT=[4.573]
1,3,0):RECHARGE=[1.514]
1,3,1):SHOOT=[4.689]
1,3,2):SHOOT=[6.251]
2,0,0):RECHARGE=[-28.809]
2,0,1):DODGE=[-26.016]
2,0,2):DODGE=[-23.754]
2,1,0):RECHARGE=[-22.49]
2,1,1):SHOOT=[-19.617]
2,1,2):SHOOT=[-16.737]
2,2,0):RECHARGE=[-16.054]
2,2,1):SHOOT=[-13.1]
2,2,2):SHOOT=[-10.137]
2,3,0):RECHARGE=[-11.272]
2,3,1):SHOOT=[-8.257]
2,3,2):SHOOT=[-5.233]
3,0,0):RECHARGE=[-45.556]
3,0,1):DODGE=[-42.975]
```

Part 2

We observe that it converges quicker than task 1 and even part 3.
This is because the gamma value is very less as compared to task 1 and even the step cost for shooting.
As the gamma value is lesser, the change in state utility value will reach less updation(magnitude of change in state's utility value, the summation part below), quicker as compared to task 1 and hence the maximum change in consecutive iterations will be lesser than bellman error earlier than task 1 and hence converge will be achieved faster.
It is achieved faster as compared to part 3 as the bellman error is larger and hence it will reach it earlier than 10^-10.
We observe that it is very less iterations as compared to Part 1 due to the huge difference in the gamma value(the discount factor).

$$R(I|A) + \gamma \Sigma_J P(J|I,A) * U_t(J)$$



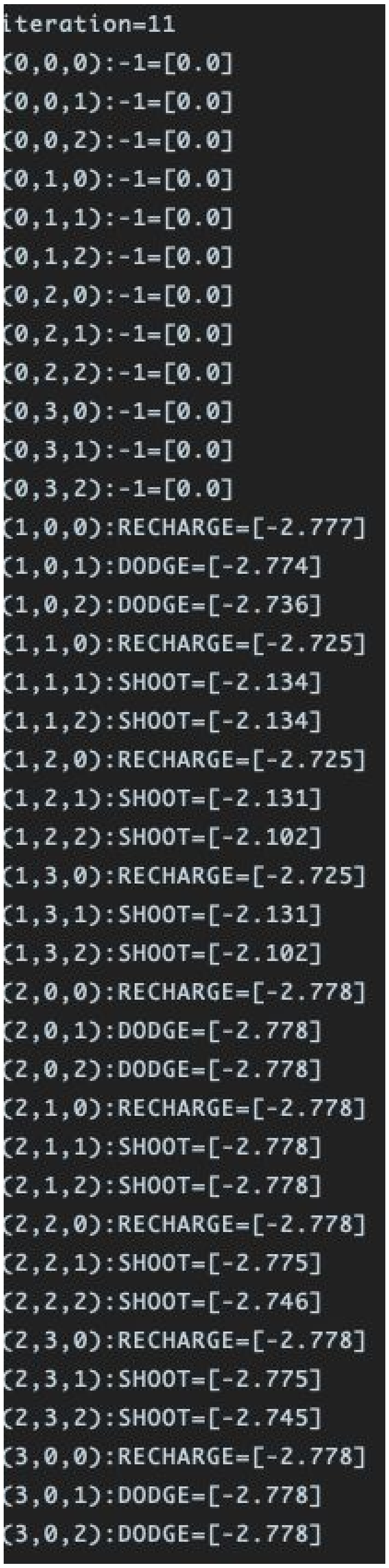
```
iteration=5
(0,0,0):-1=[0.0]
(0,0,1):-1=[0.0]
(0,0,2):-1=[0.0]
(0,1,0):-1=[0.0]
(0,1,1):-1=[0.0]
(0,1,2):-1=[0.0]
(0,2,0):-1=[0.0]
(0,2,1):-1=[0.0]
(0,2,2):-1=[0.0]
(0,3,0):-1=[0.0]
(0,3,1):-1=[0.0]
(0,3,2):-1=[0.0]
(1,0,0):RECHARGE=[-2.777]
(1,0,1):DODGE=[-2.774]
(1,0,2):DODGE=[-2.736]
(1,1,0):RECHARGE=[-2.725]
(1,1,1):SHOOT=[-2.134]
(1,1,2):SHOOT=[-2.134]
(1,2,0):RECHARGE=[-2.725]
(1,2,1):SHOOT=[-2.131]
(1,2,2):SHOOT=[-2.102]
(1,3,0):RECHARGE=[-2.725]
(1,3,1):SHOOT=[-2.131]
(1,3,2):SHOOT=[-2.102]
(2,0,0):RECHARGE=[-2.778]
(2,0,1):RECHARGE=[-2.778]
(2,0,2):DODGE=[-2.778]
(2,1,0):RECHARGE=[-2.778]
(2,1,1):SHOOT=[-2.778]
(2,1,2):SHOOT=[-2.778]
(2,2,0):RECHARGE=[-2.778]
(2,2,1):SHOOT=[-2.775]
(2,2,2):SHOOT=[-2.746]
(2,3,0):RECHARGE=[-2.778]
(2,3,1):SHOOT=[-2.775]
(2,3,2):SHOOT=[-2.745]
(3,0,0):RECHARGE=[-2.778]
(3,0,1):RECHARGE=[-2.778]
(3,0,2):RECHARGE=[-2.778]
```

Part 3

We observe that the number of iterations increase as compared to part 2 as even though the gamma i.e discount factor is the same the bellman error is very lesser and hence it will converge in a little more iterations as compared to part 2. But not by a lot as discount factor is 10^-1 and hence they multiply in successive iterations reaching a less than comparable with respect to the belmann error of 10^-10.

We observe that it is very less iterations as compared to Task 1 due to the huge difference in the gamma value(the discount factor).
The same can be argued about the Part1.

The increase in required precision (delta) does nothing to change the inaccuracy caused due to the decrease in future weights.



```
iteration=11
(0,0,0):-1=[0.0]
(0,0,1):-1=[0.0]
(0,0,2):-1=[0.0]
(0,1,0):-1=[0.0]
(0,1,1):-1=[0.0]
(0,1,2):-1=[0.0]
(0,2,0):-1=[0.0]
(0,2,1):-1=[0.0]
(0,2,2):-1=[0.0]
(0,3,0):-1=[0.0]
(0,3,1):-1=[0.0]
(0,3,2):-1=[0.0]
(1,0,0):RECHARGE=[-2.777]
(1,0,1):DODGE=[-2.774]
(1,0,2):DODGE=[-2.736]
(1,1,0):RECHARGE=[-2.725]
(1,1,1):SHOOT=[-2.134]
(1,1,2):SHOOT=[-2.134]
(1,2,0):RECHARGE=[-2.725]
(1,2,1):SHOOT=[-2.131]
(1,2,2):SHOOT=[-2.102]
(1,3,0):RECHARGE=[-2.725]
(1,3,1):SHOOT=[-2.131]
(1,3,2):SHOOT=[-2.102]
(2,0,0):RECHARGE=[-2.778]
(2,0,1):DODGE=[-2.778]
(2,0,2):DODGE=[-2.778]
(2,1,0):RECHARGE=[-2.778]
(2,1,1):SHOOT=[-2.778]
(2,1,2):SHOOT=[-2.778]
(2,2,0):RECHARGE=[-2.778]
(2,2,1):SHOOT=[-2.775]
(2,2,2):SHOOT=[-2.746]
(2,3,0):RECHARGE=[-2.778]
(2,3,1):SHOOT=[-2.775]
(2,3,2):SHOOT=[-2.745]
(3,0,0):RECHARGE=[-2.778]
(3,0,1):DODGE=[-2.778]
(3,0,2):DODGE=[-2.778]
```

CONCLUSION

We understand the value iteration algorithm much better after this assignment and also understand an important part of machine learning.

REFERENCES

Online resources
AIMA