team_87_report

March 7, 2020

1 Team 87

Members: Jaidev Shriram (2018101012), Rohan Grover (2018101017)

This is the report for assignment-2 of team 87.

1.1 Task 1

The policy in value iteration at time t is calculated based on the utility at time t+1.

This image below shows the policy taken at each state and the corresponding utility:

This was calculated using the following formula for utility:

Here are a few observations of the policy recommended by value iteration at the end of 125 iterations.

1. Lero tries to bring down the dragon carefully. He does not prefer to shoot whenever possible as the probability of hitting is low. Hence, for instance, when the arrow count is 1, and stamina is only 50, he prefers to recharge. This would put him in a better position to shoot and dodge to hit the dragon and obtain a new arrow.

2. Lero is not a rash decision maker. Possibly due to the high value of gamma, the hero is free to prioritize long term gains rather than just shooting constantly. This is clear from the fact that lero doesn't shoot whenever possible, but chooses to dodge and recharge selectively as these will lead to states with a better guarantee of defeating the dragon.

3. The best state for the hero to be in (apart from terminal) is when he has 3 arrows, and maximum stamina. This is obvious and clear from the utility value derived as well

4. The worst state for the hero to be in is when dragon has maximum health, and hero has 0 arrows and 0 stamina. This is once again obvious, as this is the start state and Lero is expected to move away from this as early as possible.

5. Health is a key factor in determining utility. We can clearly see a trend where as health drops, the utility increases. In fact, when most states of same health, are compared with another set of states of a different health value, the two sets barely cross into each other. This is once again because, we reward lero for killing the dragon.

6. Occasional anomalies to the previous point: It is important to note that there are slight excpetions as for instance state (2,3,2) has a higher utility than (1,0,0). This also makes sense as despite the dragon being nearly done, lero is unable to shoot, dodge, and can only wait and hope for a recharge. Even after this, he must hope for an arrow. Whereas, in the previous case, Lero has 3 arrows and maximum stamina which are essential for the mission.

```
(1,0,0):RECHARGE=[-179.508]
(1,0,1):DODGE=[-156.522]
(1,0,2):DODGE=[-136.991]
(1,1,0):RECHARGE=[-127.499]
(1,1,1):SHOOT=[-103.856]
(1,1,2):SHOOT=[-92.478]
(1,2,0):RECHARGE=[-102.076]
(1,2,1):SHOOT=[-78.112]
(1,2,2):SHOOT=[-66.409]
(1,3,0):RECHARGE=[-89.648]
(1,3,1):SHOOT=[-65.527]
(1,3,2):SHOOT=[-53.665]
(2,0,0):RECHARGE=[-351.231]
(2,0,1):DODGE=[-330.413]
(2,0,2):DODGE=[-312.724]
(2,1,0):RECHARGE=[-304.128]
(2,1,1):RECHARGE=[-282.716]
(2,1,2):SHOOT=[-261.033]
(2,2,0):RECHARGE=[-255.68]
(2,2,1):SHOOT=[-233.656]
(2,2,2):SHOOT=[-211.353]
(2,3,0):RECHARGE=[-219.569]
(2,3,1):SHOOT=[-197.089]
(2,3,2):SHOOT=[-174.325]
(3,0,0):RECHARGE=[-506.755]
(3,0,1):DODGE=[-487.901]
(3,0,2):DODGE=[-471.881]
(3,1,0):RECHARGE=[-464.096]
(3,1,1):DODGE=[-444.703]
(3,1,2):SHOOT=[-425.066]
(3,2,0):RECHARGE=[-420.218]
(3,2,1):SHOOT=[-400.271]
(3,2,2):SHOOT=[-380.072]
(3,3,0):RECHARGE=[-375.086]
(3,3,1):SHOOT=[-354.569]
(3,3,2):SHOOT=[-333.794]
(4,0,0):RECHARGE=[-647.604]
(4,0,1):DODGE=[-630.529]
(4,0,2):DODGE=[-616.021]
(4,1,0):RECHARGE=[-608.97]
(4,1,1):DODGE=[-591.407]
(4,1,2):SHOOT=[-573.623]
(4,2,0):RECHARGE=[-569.232]
(4,2,1):SHOOT=[-551.167]
(4,2,2):SHOOT=[-532.874]
(4,3,0):RECHARGE=[-528.358]
(4,3,1):SHOOT=[-509.777]
(4,3,2):SHOOT=[-490.962]
```

policy_task1.png

$$\Pi_{t+1} = \arg\max_{A} R(I|A) + \gamma \Sigma_J P(J|I,A) U_{t+1}(J)$$

policy_formula.png

```
(1,0,0):RECHARGE=[-14.434]
(1,0,1):DODGE=[-11.459]
(1,0,2):DODGE=[-8.932]
(1,1,0):RECHARGE=[-7.704]
(1,1,1):SHOOT=[-4.645]
(1,1,2):SHOOT=[-3.172]
(1,2,0):RECHARGE=[-4.414]
(1,2,1):SHOOT=[-1.313]
(1,2,2):SHOOT=[0.201]
(1,3,0):RECHARGE=[-2.806]
(1,3,1):SHOOT=[0.315]
(1,3,2):SHOOT=[1.85]
(2,0,0):RECHARGE=[-36.654]
(2,0,1):DODGE=[-33.96]
(2,0,2):DODGE=[-31.672]
(2,1,0):RECHARGE=[-30.559]
(2,1,1):SHOOT=[-27.789]
(2,1,2):SHOOT=[-24.983]
(2,2,0):RECHARGE=[-24.29]
(2,2,1):RECHARGE=[-21.44]
(2,2,2):SHOOT=[-18.554]
(2,3,0):RECHARGE=[-19.618]
(2,3,1):SHOOT=[-16.709]
(2,3,2):SHOOT=[-13.763]
(3,0,0):RECHARGE=[-56.778]
(3,0,1):DODGE=[-54.338]
(3,0,2):DODGE=[-52.265]
(3,1,0):RECHARGE=[-51.258]
(3,1,1):DODGE=[-48.749]
(3,1,2):SHOOT=[-46.208]
(3,2,0):RECHARGE=[-45.58]
(3,2,1):DODGE=[-42.999]
(3,2,2):SHOOT=[-40.386]
(3,3,0):RECHARGE=[-39.741]
(3,3,1):SHOOT=[-37.086]
(3,3,2):SHOOT=[-34.398]
(4,0,0):RECHARGE=[-74.997]
(4,0,1):DODGE=[-72.789]
(4,0,2):DODGE=[-70.912]
(4,1,0):RECHARGE=[-70.0]
(4,1,1):SHOOT=[-67.728]
(4,1,2):SHOOT=[-65.427]
(4,2,0):RECHARGE=[-64.859]
(4,2,1):SHOOT=[-62.522]
(4,2,2):SHOOT=[-60.156]
(4,3,0):RECHARGE=[-59.571]
(4,3,1):SHOOT=[-57.167]
(4,3,2):SHOOT=[-54.733]
```

policy_2_1.png

1.2 Task 2

For this task, step cost was set to -2.5

1.2.1 Part 1

The number of iterations have dropped from 125 to 103 here. The policy may be viewed here:

Now that the cost of shooting is much lower, we see an increase in the number of shoot actions taken. In fact, some dodges and recharges have been replaced by a SHOOT action. This is probably because it is cheap to shoot. Shooting in turn has two possibilities, reducing the life of the dragon, which is clearly beneficial, or missing the target. Missing the target has the cost of a lost arrow and a lower stamina, but there is barely any additional cost due to shooting. Hence, in the long term, it is acceptable to shoot instead of recharging when the life of the dragon is high. Hence, for health >=3, there are more shoots as the distance from terminal state is significant.

Mathematically, this is backed up as the cost of shoot is low (10 times lower than normal). Hence, the action taken would primarily depend on the utility of the resulting state, rather than on the cost of taking the action itself.

```
(1,0,0):RECHARGE=[-2.775]
(1,0,1):DODGE=[-2.744]
(1,0,2):DODGE=[-2.441]
(1,1,0):RECHARGE=[-2.358]
(1,1,1):SHOOT=[2.361]
(1,1,2):SHOOT=[2.363]
(1,2,0):RECHARGE=[-2.357]
(1,2,1):SHOOT=[2.382]
(1,2,2):SHOOT=[2.618]
(1,3,0):RECHARGE=[-2.357]
(1,3,1):SHOOT=[2.382]
(1,3,2):SHOOT=[2.619]
(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=[-2.776]
(2,2,0):RECHARGE=[-2.776]
(2,2,1):SHOOT=[-2.757]
(2,2,2):SHOOT=[-2.521]
(2,3,0):RECHARGE=[-2.776]
(2,3,1):SHOOT=[-2.757]
(2,3,2):SHOOT=[-2.519]
(3,0,0):RECHARGE=[-2.778]
(3,0,1):DODGE=[-2.778]
(3,0,2):DODGE=[-2.778]
(3,1,0):RECHARGE=[-2.778]
(3,1,1):SHOOT=[-2.778]
(3,1,2):SHOOT=[-2.778]
(3,2,0):RECHARGE=[-2.778]
(3,2,1):SHOOT=[-2.778]
(3,2,2):SHOOT=[-2.778]
(3,3,0):RECHARGE=[-2.778]
(3,3,1):SHOOT=[-2.778]
(3,3,2):SHOOT=[-2.777]
(4,0,0):RECHARGE=[-2.778]
(4,0,1):DODGE=[-2.778]
(4,0,2):DODGE=[-2.778]
(4,1,0):RECHARGE=[-2.778]
(4,1,1):SHOOT=[-2.778]
(4,1,2):SHOOT=[-2.778]
(4,2,0):RECHARGE=[-2.778]
(4,2,1):SHOOT=[-2.778]
(4,2,2):SHOOT=[-2.778]
(4,3,0):RECHARGE=[-2.778]
(4,3,1):SHOOT=[-2.778]
(4,3,2):SHOOT=[-2.778]
```

policy_2_2.png

1.2.2 Part 2

The number of iterations have drastically dropped from 100's to 4. The policy may be viewed here:

The role of gamma here is to determine a stopping point for the algorithm. With a lower gamma, short term gains are prioritized and lero attempts to kill the dragon sooner. Since the long term is not considered, a part of the decision making for states farther from the terminal is seemingly random. This is because lero is unable to plan ahead.

If the value of gamma was higher, it would be able to refine between multiple iterations, and make more decisions about an optimum policy as the utility value of states do not drop as drastically.

1.2.3 Part 3

The number of iterations have increased from 4 to 11. The policy may be viewed here

Here once again, future rewards are trivial due to low gamma value and the results of the

```
(1,0,0):RECHARGE=[-2.775]
(1,0,1):DODGE=[-2.744]
(1,0,2):DODGE=[-2.441]
(1,1,0):RECHARGE=[-2.358]
(1,1,1):SHOOT=[2.361]
(1,1,2):SHOOT=[2.363]
(1,2,0):RECHARGE=[-2.357]
(1,2,1):SHOOT=[2.382]
(1,2,2):SHOOT=[2.618]
(1,3,0):RECHARGE=[-2.357]
(1,3,1):SHOOT=[2.382]
(1,3,2):SHOOT=[2.619]
(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=[-2.776]
(2,2,0):RECHARGE=[-2.776]
(2,2,1):SHOOT=[-2.757]
(2,2,2):SHOOT=[-2.521]
(2,3,0):RECHARGE=[-2.776]
(2,3,1):SHOOT=[-2.757]
(2,3,2):SHOOT=[-2.519]
(3,0,0):RECHARGE=[-2.778]
(3,0,1):DODGE=[-2.778]
(3,0,2):DODGE=[-2.778]
(3,1,0):RECHARGE=[-2.778]
(3,1,1):SHOOT=[-2.778]
(3,1,2):SHOOT=[-2.778]
(3,2,0):RECHARGE=[-2.778]
(3,2,1):SHOOT=[-2.778]
(3,2,2):SHOOT=[-2.778]
(3,3,0):RECHARGE=[-2.778]
(3,3,1):SHOOT=[-2.778]
(3,3,2):SHOOT=[-2.777]
(4,0,0):RECHARGE=[-2.778]
(4,0,1):DODGE=[-2.778]
(4,0,2):DODGE=[-2.778]
(4,1,0):RECHARGE=[-2.778]
(4,1,1):SHOOT=[-2.778]
(4,1,2):SHOOT=[-2.778]
(4,2,0):RECHARGE=[-2.778]
(4,2,1):SHOOT=[-2.778]
(4,2,2):SHOOT=[-2.778]
(4,3,0):RECHARGE=[-2.778]
(4,3,1):SHOOT=[-2.778]
(4,3,2):SHOOT=[-2.778]
```

policy_2_3.png

previous part remain true. However here, due to the exit condition of the value iteration algorithm which requires that the maximum difference in utility between the previous iteration and current one is less than delta, and here delta is much smaller. This gives us more precision about the policy. However, this will still perform badly as any insight into the future is lost due to low gamma value.