OPTIMAL MANIPULATION OF REWARDS IN REINFORCEMENT LEARNING

CHARANTEJA SAKAMURI	1530974
DEVARSH DANI	1516908
RAKSHIT VALLABHANENI	1499487

CHAPTER 1

1.1 Introduction

Reinforcement Learning is an area of machine learning that tries to learn about an environment by interacting with it. Reinforcement learning is centered around the idea of a reward function that indicates the learning algorithm what states of the environment are to move to or avoid. These rewards basically provide necessary hints to the learning algorithm to help understand the environment and the best course of action. The policy determines the best course of action that can be taken at a particular state. The agent who is trying to reach the goal state receives a reward or penalty depending on the efficiency of action. The efficiency of a state is measured in terms of utility. The good states which would be beneficial to the agent and help reach the goal will have higher utilities to encourage the agent to take those paths and the bad states will have penalties to discourage the agent from taking them. As the agent increases this interaction with the environment, it begins taking more promising paths frequently. A learning agent effectively tries to sense the state of its environment and take actions that affect the state.

Reinforcement learning is very different to supervised learning, where we learn from a training set of labeled examples provided by a knowledgeable external supervisor. This type of learning is found to be very effective for some cases where training sets can somehow resemble the patterns and trends present in the actual data. This is not the case most of the times as in interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act.

Another field of machine learning called unsupervised learning also falls short in some cases where reinforcement learning can do better. Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. It might seem that reinforcement learning is

a type of unsupervised learning owing to the fact that it does not rely on examples of correct behavior, reinforcement learning is inherently different because it tries maximize a reward instead of trying to find a hidden structure.

The main feature that makes reinforcement learning successful is that it explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment.

1.2 ELEMENTS

Reinforcement Learning consists of four main elements and most of the solutions revolve around these four elements.

1. **Policy**

   The policy essentially determines the learning agent's way of behaving at a given time. A policy can be said to be a mapping from perceived states of the environment to actions to be taken when in those states.

2. **Reward**

   The Reward is responsible for helping the agent understand if the action it is trying to make is a correct one. On each step, the environment provides the agent a reward (if negative, it may be called as penalty) and the agents sole objective is to maximize the total reward it receives over the long run. It basically helps in determining good and bad events for the agent.

3. **Value Function**

   The reward usually indicates to the agent what maybe a good even in the immediate or current scenario. A value function, specifies what is good in the long run. The value of a state can be said to be the total amount of reward an agent can expect to accumulate over the future, starting from that state. Reward determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into

account the states that are likely to follow, and the rewards available in those states.

4. **Model**

   Model tries to mimic the behavior of the environment which helps us to make inferences about how the environment will behave. Model-based methods are those that make use of these models in order to solve reinforcement learning problems as opposed to model-free methods that are explicitly trial-and-error learners.

1.3 Q-Learning

Q-Learning is an off-policy temporal difference control algorithm that is model-free. Its working principle is that it tries to learn an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy from then onwards.

The Utility of a state is calculated using the formula:

$$Q(a,s) \leftarrow (1-\alpha)*Q(a,s) + \alpha*[R(s',a,s)+ \gamma*\max_{a'}Q(a',s')]$$

where

Q is the current utility, initially zero.

α is the learning rate.

s is the current state being evaluated.

R is the immediate reward function for reaching state s' by taking action 'a' in state 's'.

γ is the discount rate.

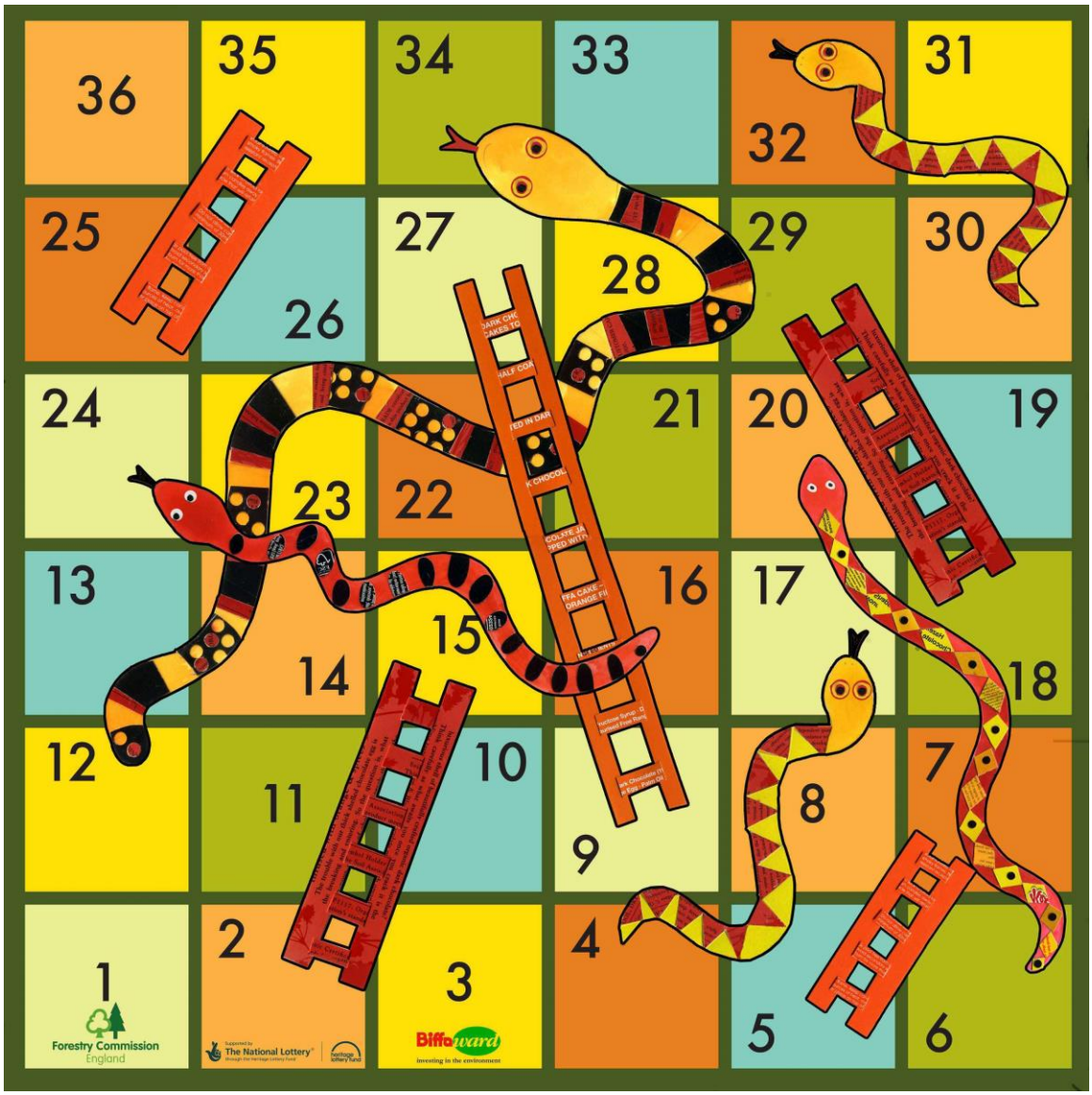
$\max_{a'}Q(a',s')$ is the max utility obtainable from all valid actions after taking action 'a' in state 's'.

Here we can see that the learned action-value function Q directly approximates the optimal action-value function. The agent always tries to move towards states with higher utility values with the assumption that it leads to the goal state enabling early convergence.

CHAPTER 2

2.1 Environment Description

The environment the agent is learning is Snakes and Ladders game. Below is the image of the game.



2.2 Why Snakes and Ladders Game?

The Game of Snakes and Ladders provides us a representative environment of how most of the problems that can be solved by reinforcement learning look like. The main idea behind the game is that a player has to reach the goal state in order to win. Usually 2 or more players participate in the game, at each turn each player gets to roll a dice and the result of the dice indicates the number of steps the player can move. The environment contains numerous ladders that can help you jump up a particular number of states towards a goal state and snakes that move you away from the goal state. The player who reaches the goal state in the shortest number of turns is the winner.

2.3 Formulating Snake and Ladders as a Reinforcement Learning Problem:

The environment usually consists of several states and the agent can be a part of any one of the states. The agent tries to take one of several available actions to reach the goal state, these actions are determined by the policy.

In our problem, each position number is considered as a state. So, there are 36 states numbered from 1 through 36. And at each state the agent has 8 possible actions (make a move 1 step-6 steps, climb the ladder, bitten by the snake). Since climbing a ladder or getting bitten by a snake changes the state of the agent thus they are also considered as actions.

This game can be implemented using Reinforcement learning, where the agent initially learns about the environment and then formulates an Optimal Path to reach the goal state. Note after the learning phase agent is made to start from position 1 and it tries to reach the goal state with the help of Q table. Q table can be thought of as a meta learning result that helps the agent to take decisions based on its current state.

The agent starts from the position numbered 1 in the above board. The goal of the agent is to reach position numbered 36. We have ladders to help the agent reach the goal state faster. For this reason, we are giving positive rewards for the states where ladder is present to encourage the agent to take these states. The obstacles here are the snakes which pull the agent back, so negative rewards are given to these states to discourage the agent from taking these paths. The magnitude of reward is proportional to the number of steps you go ahead using a ladder or number of steps you go back using the snake.

2.4 Algorithm:

1. The Algorithm Consists of 2 phases:
   i) Randomly **exploring** new states in the environment
   Initially, when the agent doesn't know anything about the environment. It randomly explores new states depending on the roll of a dice which in our case is a random number between 1 through 6. And updates the Q table.
   ii) Greedily **exploiting** the learnt environment.
   When the agent has sufficiently learnt about the environment. It exploits the environment based on the learnt Q table. Reinforcement learning is a continuous learning process in which the agent tries to improve its knowledge about the environment each time it interacts.

For each of the above two phases we do the following:
Perform goal test:
If goal is reached, then restart the game.
else if we have a ladder in the current state, climb the ladder
else if we have snake in current state then agent gets bitten by it and is pulled down.
else roll a die and act depending on the die result this is for phase 1.
For phase 2 we take that action which has the maximum Q(s,a) value. A can go through 1 to 6.

2.5 Observations

Q table:

State/Action	1	2	3	4	5	6	Ladder	Snake
36	0	0	0	0	0	0	0	0
35	-0.99725	-0.657	-0.71613	-0.96176	-0.327	-0.71613	0	0
34	0	0	0	0	0	0	0	-0.88235
33	-6.6	-6.6	-0.8609	-0.657	-0.327	-0.71613	0	0
32	0	0	0	0	0	0	0	-0.88235
31	-0.6	-0.6	-6.6	-11.22	-0.83193	0	0	0
25	0	0	0	0	0	0	-0.83193	0
26	0	0	-0.3	0	0	0	0	0
27	-0.51	-0.32943	-0.56003	-0.3	-0.6	-0.6	0	0
28	0	0	0	0	-0.327	-6.6	0	0
29	-0.87616	-0.89007	-1.167	-0.8499	-6.8289	-0.9068	0	0
30	-0.71162	-0.6	-0.6	-6.6	-0.71613	-0.657	0	0
24	0	0	0	0	0	0	0	0
23	0	0	3	0	0	0	0	0
22	-0.3	0	0	0	0	0	0	0
21	0	0	0	3	0	-0.3	0	0
20	0	0	0	0	0	0	0	-0.51
19	-0.327	0	-0.3	0	0	7.599	0	0
13	-0.327	0	-0.327	0	0	0	0	0
14	-0.327	-0.327	0	0	3.3	0	0	0
15	0	0	10.89342	-0.03596	-4.2	-4.2	0	0
16	0	-8.541	0	0	-0.3	0	0	0
17	0	0	0	0	0	0	0	-0.3
18	0	0	0	0	0	0	-0.99944	0
12	0	-0.327	-0.327	0	-0.327	9.705861	0	0
11	-0.327	0	0	0	-0.327	-0.327	0	0
10	0	0	0	0	0	-0.327	0	0
9	0	0	0	0	0	0	-0.91765	0
8	-0.327	0	0	-0.327	0	0	0	0
7	0	0	0	-0.5559	0	-0.58803	0	0
1	12.97883	-0.17646	-0.51	-0.5559	-0.327	-0.327	0	0
2	0	0	0	0	0	0	-0.99837	0
3	0	0	0	0	-0.3459	14.97474	0	0
4	-0.327	0	0	-0.3459	-0.327	0	0	0
5	0	0	0	0	0	0	-0.657	0
6	0	0	0	-0.327	-0.327	-0.3459	0	0

From the above table we can observe the following:

1. At state 34, we can see in the Q table that the only action available in that state is snake. This snake action pulls down the agent by 22 steps to state 12. So, the reward is essentially negative indicating that agent is going to receive a penalty by moving into that particular state.
2. Similarly, at state 25 have a ladder which moves up the agent to state number 34 i.e. 9 steps ahead. We can notice from the Q table we get a positive value implying that the agent will be beneficial if it moves to that state.

Epoch	Number of Steps to reach Terminal State
1	6
2	4
3	8
4	7
5	5
6	4
7	12
8	4
9	4
10	4
11	4
12	4
13	21
14	4
15	6
16	4
17	7
18	5
19	4
20	4
21	4

The agent reached the Terminal state for 14 times using the existing algorithm. The number of steps it takes each time is uneven, indicating that it is not learning properly.

2.5.1 Flaws with the existing Reinforcement Algorithm:

Careful observation of the above Q-table reveals that the following information:

1) In this Implementation the reward for the immediate state alone is taken into consideration when calculating the Q value of a state. But the utility of a state action pair can also be dependent on the neighboring states. This is not taken into consideration in this implementation.
2) In this algorithm, the states having climb and snake bite action is not being given enough importance. It is vital for success that we try to avoid the snake bite action and take the ladder action if it is beneficial.

2.6 New Reinforcement Algorithm:

In the above algorithm, the reward that the agent gets when it travels from one state to another depends on the state in which it is moving to. Snakes and Ladders is a special kind of game in which the agent should travel by taking the ladders more frequently and avoiding the snakes.

Taking this aspect of the game into consideration, we try moving the agent in such a way that it reaches a higher state by making use of the ladders and avoiding a snake bite that pulls it back. So, we try to consider the reward of the next 4 steps and have taken a weighted average, where a state with a snake will have high weight. This helps the agent to understand and learn which steps to avoid in order to not be eaten by a snake. This substitution in the algorithm has given us the below results for our Q Table.

The new reward function –

For M in state s to state s+4

If state M contains a snake give it weight 0.6

Else if state M contains a ladder give it weight 0.25

Else give it 0.15

finalReward += currentReward

end

2.6.1 New Q Table

	Action							
State	1	2	3	4	5	6	Ladder	Snake
36	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0
33	-3.9	-6.63	-10.8151	-8.541	-3.9	-8.541	0	0
32	0	0	0	0	0	0	0	-12.7429
31	-4.2	-4.2	-4.2	-7.14	-9.198	0	0	0
25	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0
27	0	0	0	0	0	-0.351	0	0
28	0	0	-0.3	0	-0.651	0	0	0
29	-0.678	-0.3	-0.3	-0.3	0	-0.3	0	0
30	-7.14	-7.14	-7.491	-9.198	-7.14	-9.198	0	0
24	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
22	0	0	0	0	-0.3	-0.3	0	0
21	0	0	0	0	-0.3	-0.3	0	0
20	0	0	0	0	0	0	0	-2.628
19	-2.4	0	0	0	0	-2.4	0	0
13	-2.1	0	-2.1	0	0	0	0	0
14	-1.2	-1.2	0	-1.2	0	0	0	0
15	0	-1.2	0	-1.2	-1.2	-1.2	0	0
16	-3.6	-7.884	-3.6	-3.6	-3.6	-6.12	0	0
17	0	0	0	0	0	0	0	-6.98372
18	0	0	0	0	0	0	-6.65544	0
12	0	0	0	0	0	0	0	0
11	0	0	0	0	-0.324	0	0	0
10	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
8	0.327	0	0	0.3	0	0	0	0
7	0	0	0	1.998404	0	1.02	0	0
1	0	0.3	0.51	0.996705	0	0.381	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	4.08E-04	0	0	0	5.34E-04	0	0	0
5	0	0	0	0	0	0	3.998436	0
6	0	0	0.9	0	1.53	0	0	0

This Q Table theoretically considers the next few steps after the destination state as well in assigning the Q value. We hoped that this would give us better results because the agent is able to take the immediate states after the destination into consideration. These are the number of steps the new algorithm is taking up to arrive at the goal state.

Epoch	Number of Steps to reach Terminal State
1	9
2	7
3	7
4	7
5	9
6	7
7	13
8	7
9	9
10	9
11	12
12	6
13	17
14	7
15	7
16	9
17	4
18	9
19	4
20	7
21	9

2.6.2 Comparison

The red line indicates the modified algorithm and the blue line indicates the existing algorithm. The number of steps taken to reach the goal state keeps decreasing through the path of the graph because the agent is learning about the environment by interacting with it. We see a spike between iteration 10-15 because the agent has started encountering snake bites and then starts decreasing when it learns that it must avoid these states.

CONCLUSION

Reinforcement Learning understands the very important phenomenon of using value in the search for the goal state in the space of policies. This use of value function makes it unique from other evolutionary methods that search directly in the policy space guided by scalar evaluations of entire policies. It also emphasizes on learning by an agent from direct interaction with its environment, without having to rely on external supervision or relying on a model of the environment.

We have tried to improvise the algorithm for a particular set of environments where the reward of the reaching state and its neighboring states maybe a better indicator to the agent towards its goal. This may not be true in all the environments where only the current state is an important driver towards the goal state. This method may also be viable in dynamic environments where the goal-state is ever changing or moving.

The agent reaches the goal state more number of times with the existing algorithm than the modified one. We are seeing better results in existing algorithm because the weightage we are giving to the different actions in the states to calculate the reward might not be appropriate. Properly plugging in the weights may give us better results but this will require more experimentation.

References

1. Blackboard Class Presentations

2. Sutton and Barto (2017). *Reinforcement Learning: An Introduction*. Massachusetts: The MIT Press.

3. https://en.wikipedia.org/wiki/Reinforcement_learning

4. Hasselt, Guez and Silver (2016). *Deep Reinforcement Learning with Double Q-learning.*

5. Kaelbling, Littman and Moore (1996). *Reinforcement Learning: A Survey.*