Reinforcement Learning

Parameters:

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(s',a')$$

1. R(s):
For every action from current state, the robot gets an immediate walk/live reward of -0.05. This ensures that the robot doesn't keeps on trying a bad action like hitting a wall continuously.

2. Discount (γ) :

This defines the portion of the next state reward that the robot gets for a given action. This value is 0.8 (i.e. 80%) for this implementation.

3. T(s, a, s'):

T defines the transition model, which is the probability of landing in a new state from current state by taking an action a.

In deterministic learning, the probability of getting into a new state s' from the current state s by taking the action a is always 1 and is 0 for all other actions.
In stochastic learning, this probability is spread across different actions. For this implementation, probability of s' from s by taking a is 0.6 while it is 0.1 for the other three actions and 0.1 for staying at the same place.

Convergence:
The above equation gives a new Q value for a given action. When the robot is in the learning phase, a higher weight is given to the new Q values as compared to the previously learned Q value due to high uncertainty. However as the number of iteration increases, the weight of the newer Q values decreases and the weight of the older Q values increases as the uncertainty decreases over time.

The learning rate α which is a function of number of iteration controls the convergence of Q table.

$$Q(s,a) = \alpha(Q(s,a)) + (1-\alpha)(R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(s',a'))$$

where, $\alpha = n^{-0.1}$

n - Number of iterations

- The total number of episodes or iterations are limited to 10000 by default and can be changed in the program.
- In each episode, the robot starts again after 300 steps.

Exploration:
Epsilon - Probability of taking an action other than optimum action.

Choice of actions:

1. Always Greedy:
   When the value of epsilon is zero, the robot always takes an optimum action from current state. Thus the chances of staying in a local minima is high.
2. Always Random:
   When the value of epsilon is one, the robot always takes an action which is not optimum. Although the Q table will be updated with the highest reward path, the robot will always keep on exploring without moving on the highest reward path.
3. Softmax Exploration (Simulated Annealing):
   The Softmax exploration biases the exploration towards higher reward actions. Boltzmann distribution gives a fair estimate of an action in the direction of high reward action using the Q values encountered.

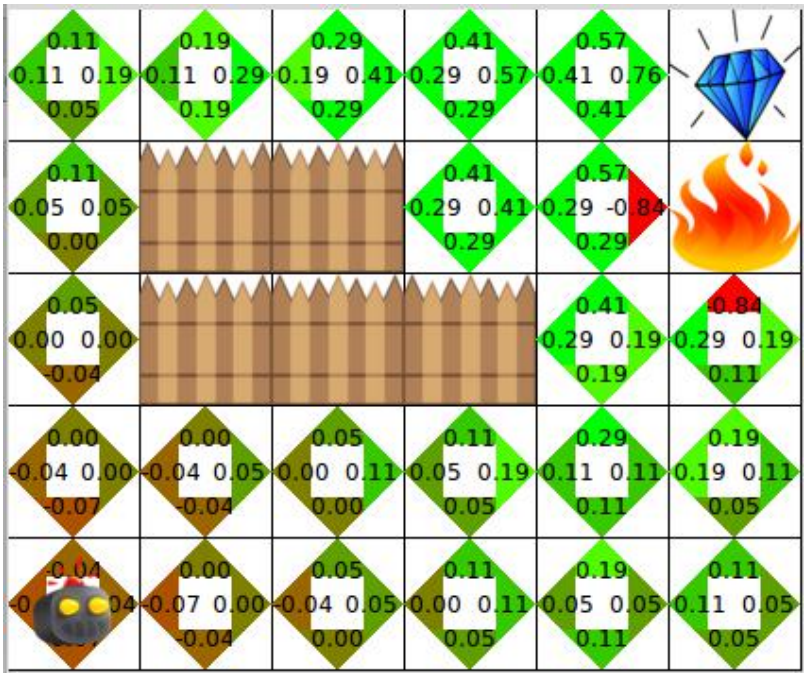
$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{e^{\sum_{a' \in \mathcal{A}} \frac{Q(s,a')}{\tau}}}$$

In the current implementation, we have a function to get the current softmax value. However, by default it is disabled to give interactive control of epsilon value to the user in the GUI. In order to switch between softmax policy and manual control epsilon, use one of the following statement in the q_learn method.

```
248     epsilon = World.w2.get()
249     # epsilon = soft_max(current, iter)
```

Heat Maps

A. Deterministic Learning

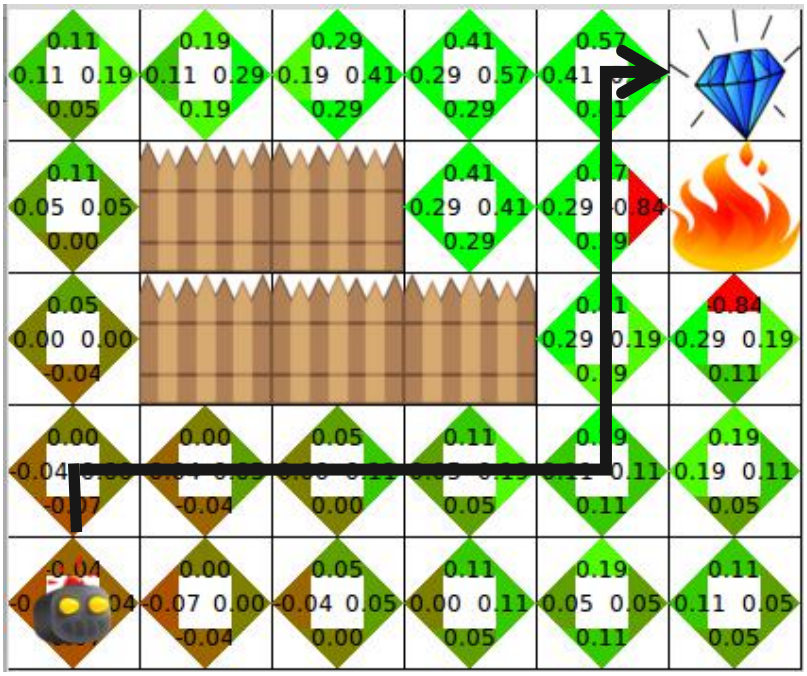


B. Stochastic Learning

Example 1

A. Deterministic Learning

The robot goes through the shortest path to get more reward, even if it has a pit alongside that path. This is because the transition model is such that the robot will never land in the pit once it has learned the Q values.



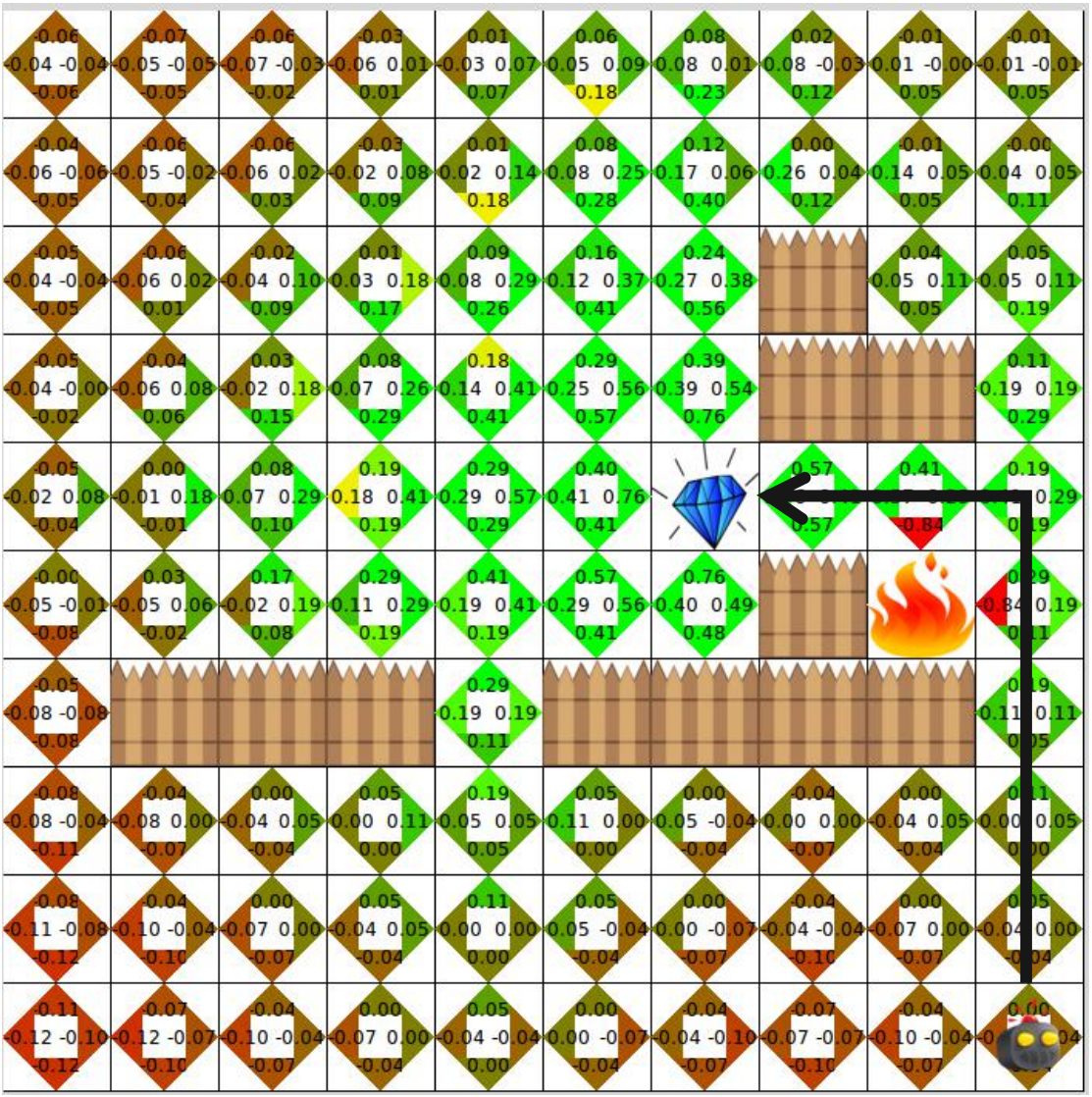
B. Stochastic Learning

We can see that for stochastic version of the above map, the robot chose a different path because the reward in the previous path (with pit) has been reduced due to the transition model effect from neighboring states.

Example 2

A. Deterministic Learning

The robot goes through the shortest path to get more reward, even if it has a pit alongside that path. This is because the transition model is such that the robot will never land in the pit once it has learned the Q values.



B. Stochastic Learning

We can see that for stochastic version of the above map, the robot chose a different path because the reward in the previous path (with pit) has been reduced due to the transition model effect from neighboring states.