Reinforcement Learning
Project 1 – Grid World using Q-Learning

A reinforcement learning system can be depicted in the following way –

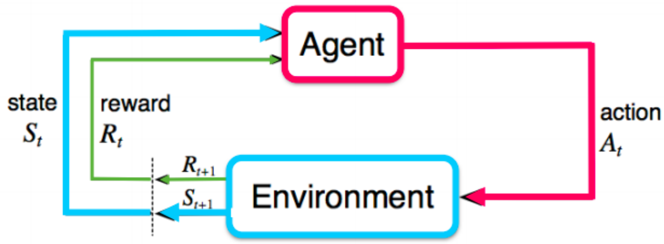

Figure 1

For the given system at hand, the environment looks like this –

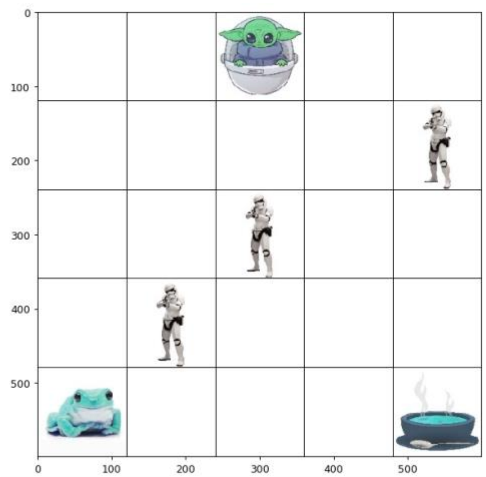

Figure 2 Grid World (5 x 5)

The system can be defined using a Markov Decision Process which is defined by tuple (S, A, P, R, γ), where

- S is the set of states, which characterizes the configuration of the environment. For the system, the set S is a 5×5 grid world environment, totaling to 25 states. Each state is represented by a $[row, column]$ tuple.
- A denotes the actions the agent can take. For the system, the agent can take 4 actions $A \in \{Up, Down, Left, Right\}$.
- R is the reward function. For the system, the reward set, $R \in \{-1, +7, +20, -7\}$.
- P is the state transition probability distribution. There are two types of environments, deterministic and stochastic.
    - For the deterministic environment, $P = 1$.
    - For the stochastic environment,

$$P = \begin{cases} 0.8, & optimal\ action \\ 0.2, & opposite\ to\ optimal\ action \end{cases}$$

- γ is the discount factor. It is initially set to 1 and decayed exponentially.

The grid world environment depicted in Figure 2 has been described using the MDP above. The initial states of the system components are –

- Initial agent state: $s[0, 2]$.
- Terminal states:
  - Goal states: $s[4,0]$ and $s[4,4]$
  - Villain states: s[1,4], s[2,3] and s[3,1]
- Rewards

$$R = \begin{cases} +20, & agent_{pos} = goal^2_{pos} \\ +5, & agent_{pos} = goal^1_{pos} \\ -10, & agent_{pos} = villain_{pos} \\ -1, & otherwise \end{cases}$$

Q-Learning

Q-learning is an off policy and model-free reinforcement learning algorithm that seeks to find the best action given the current state. The algorithm is what makes the agent "learn". The 'Q' in Q-learning stands for quality, that is, how useful a given action is gaining future reward.

It's called an off policy algorithm because the q-learning function learns from actions that are outside the current policy, like taking random or greedy actions, and therefore an explicit policy is not needed. Q-learning seeks to learn the policy that maximizes the total reward.

It's called a model-free algorithm because the q-learning function doesn't need transition probability distribution associated with the MDP to solve the environment.

Before the learning begins, we initialize the Q-table. The Q-table is a $S \times R$ dimensional table. For the given grid world environment, S is 5×5 dimensional. Hence, the Q-table is a $5 \times 5 \times 4$ dimensional table and we initialize it with zeros.

Algorithm –

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

Figure 3 Q-Learning value update algorithm

where,

$$s_t \text{ is the current state}$$
$$a_t \text{ is the action selected}$$
$$s_{t+1} \text{ is the next state}$$
$$\alpha \text{ is the learning rate}$$
$$\gamma \text{ is the discount factor}$$

At each time step t, the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1}, and Q is updated. The algorithm in its core is a Bellman Equation, using the weighted average of the old value and the new information. The distinctive feature of Q-learning algorithm is the action selected is by using a greedy policy.

The intuition behind the algorithm is that error between the ground truth and predicted value is weighted by learning rate, α. The predicted value is the Q-value for the current state. The ground truth value is the addition of the immediate reward, r_t and the future discounted reward.

Observations and Analysis

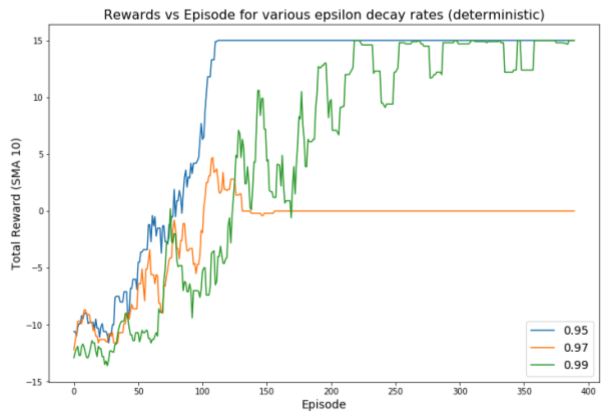

Figure 4a Change of reward per episode (Deterministic environment)

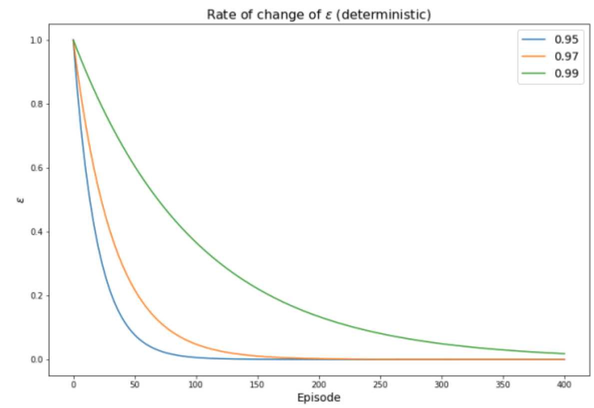

Figure 4b Change of epsilon values with episodes (Deterministic environment)

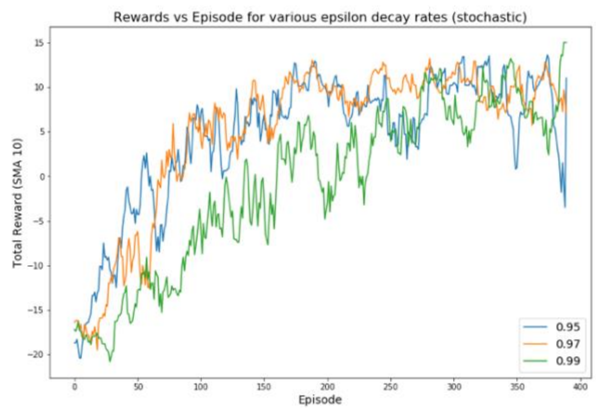

Figure 5a Change of reward per episode (Stochastic environment)

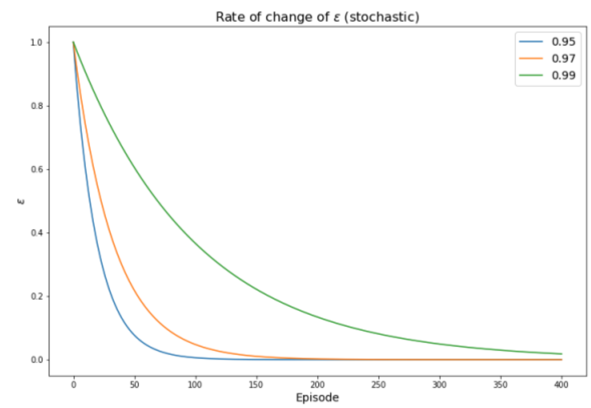

Figure 5b Change of epsilon values with episodes (Deterministic environment)

1. It can be observed that faster the convergence of epsilon, faster the agent learns the environment **given** that number of episodes are sufficiently large.
2. A complex environment will require you to train the agent for significantly large number of iterations. An example of such an environment can be if two agents block the path - $s[2,2]$ *and* $s[2,4]$.
3. **Q-learning** is more widely used than **DP** most probably due to its simplicity. You don't even need to do policy update as we can greedily choose it from the Q-values.

Results

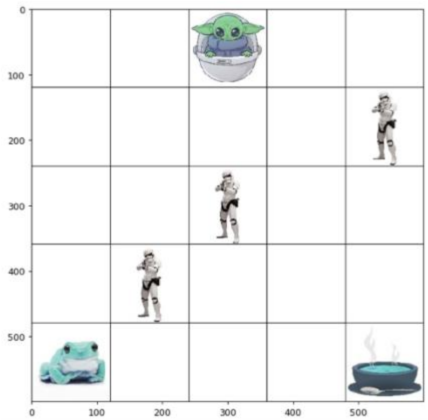

Figure 6a

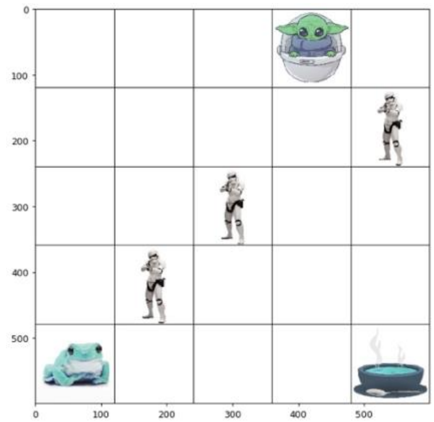

Figure 6b

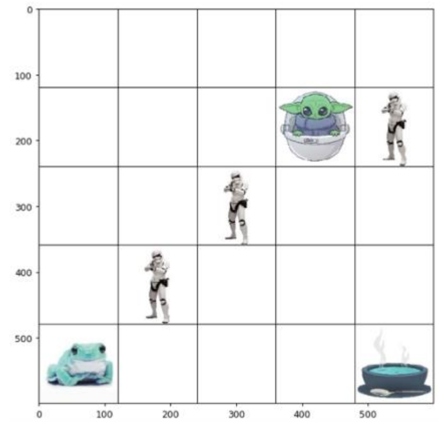

Figure 6c

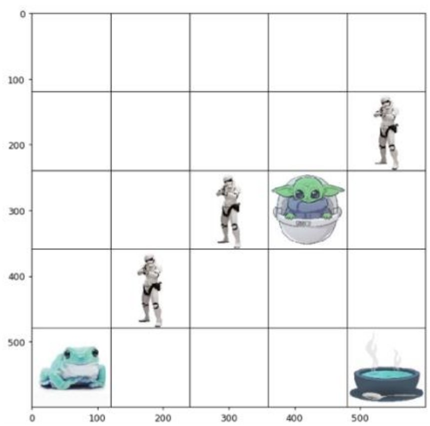

Figure 6d

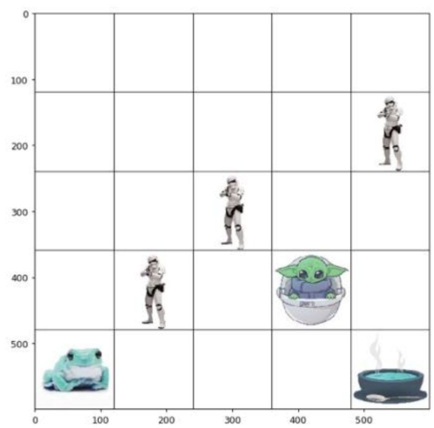

Figure 6e

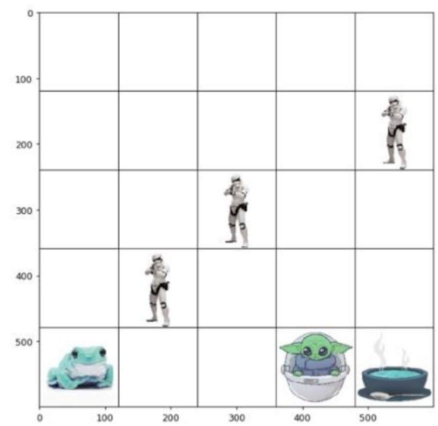

Figure 6f

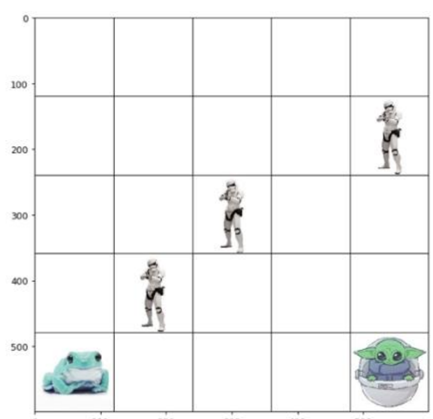

Figure 6g