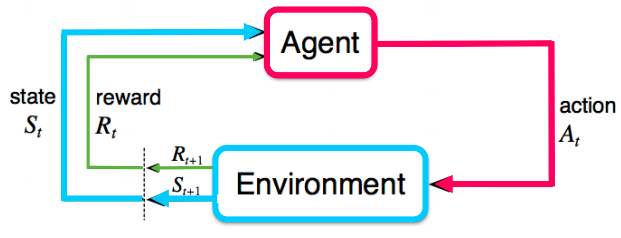
**Reinforcement Learning**

**Project 1 – Grid World using Q-Learning**

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



Figure

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

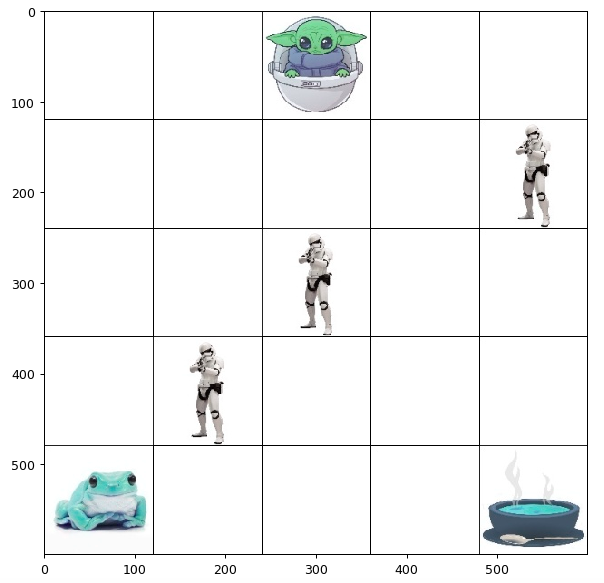


Figure Grid World (5 x 5)

The system can be defined using a Markov Decision Process which is defined by tuple ), where

* is the set of states, which characterizes the configuration of the environment. For the system, the set is a grid world environment, totaling to 25 states. Each state is represented by a tuple.
* denotes the actions the agent can take. For the system, the agent can take 4 actions .
* is the reward function. For the system, the reward set, .
* is the state transition probability distribution. There are two types of environments, deterministic and stochastic.
  + For the deterministic environment, .
  + For the stochastic environment,
* is the discount factor. It is initially set to 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: .
* Terminal states:
  + Goal states:
  + Villain states: s[1,4], s[2,3] and s[3,1]
* Rewards

**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 dimensional table. For the given grid world environment, is dimensional. Hence, the Q-table is a dimensional table and we initialize it with zeros.

**Algorithm** –

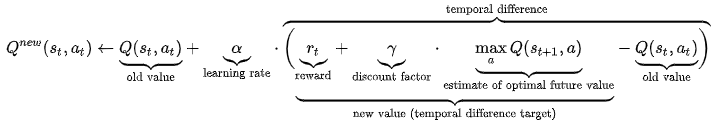


Figure Q-Learning value update algorithm

where,

At each time step , the agent selects an action , observes a reward , enters a new state , 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, . The predicted value is the Q-value for the current state. The ground truth value is the addition of the immediate reward, and the future discounted reward.

**Observations and Analysis**

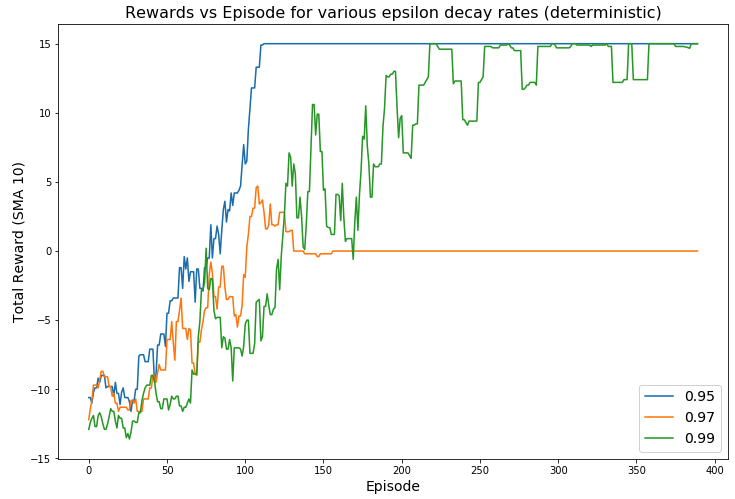
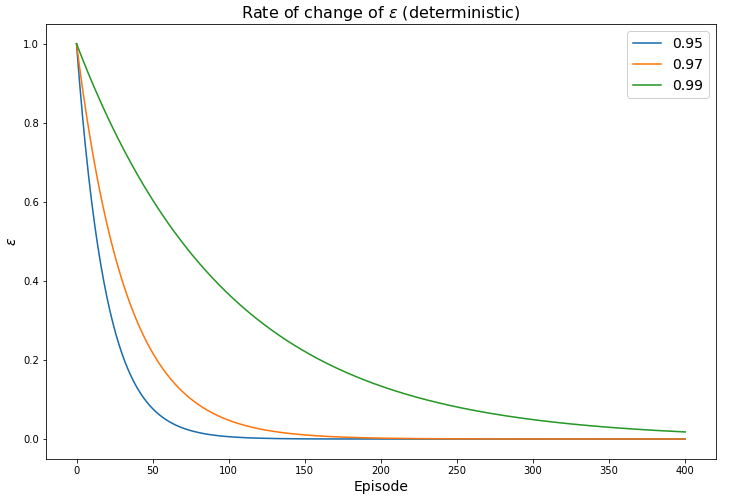


Figure 4a Change of reward per episode (Deterministic environment)

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

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

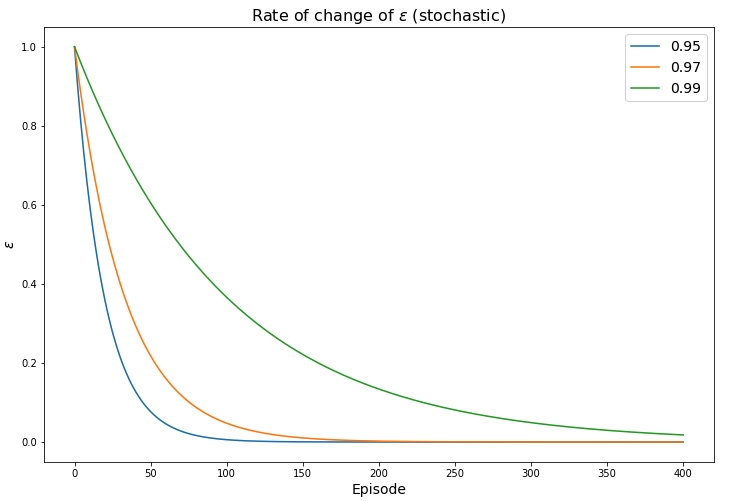
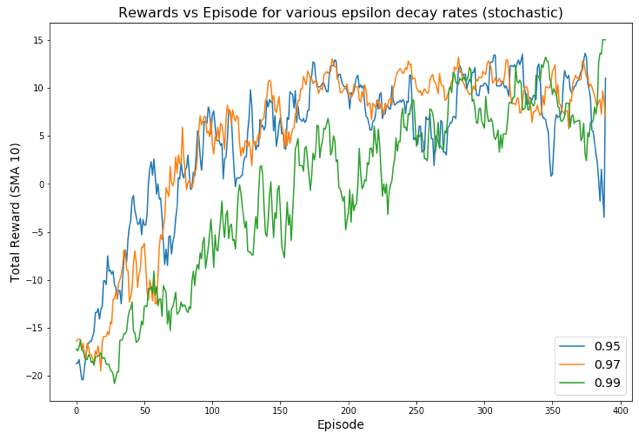


Figure 5a Change of reward per episode (Stochastic environment)

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 - .
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**

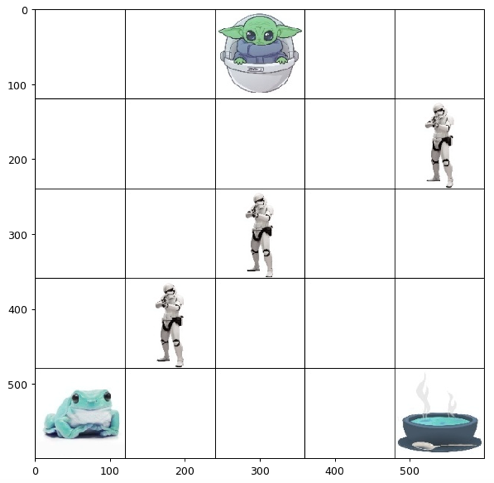
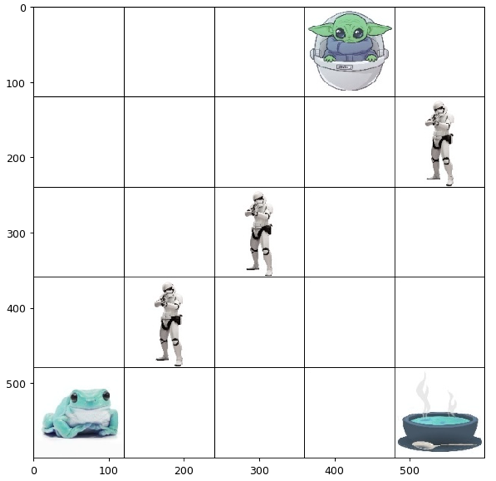
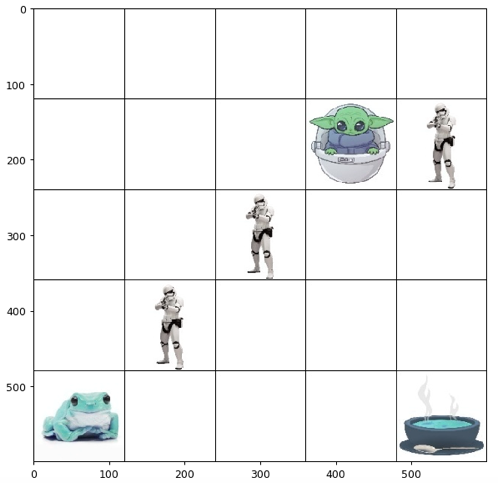


Figure c

Figure 6b

Figure 6a

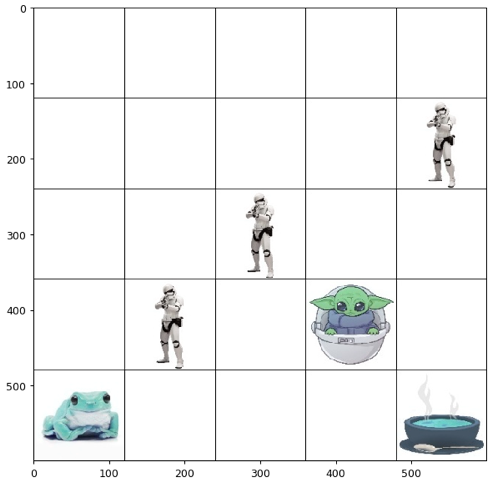
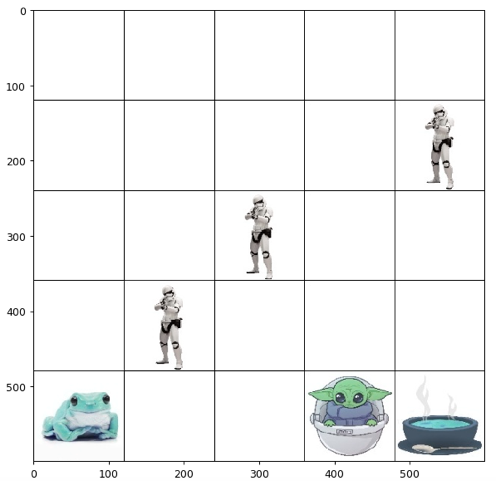
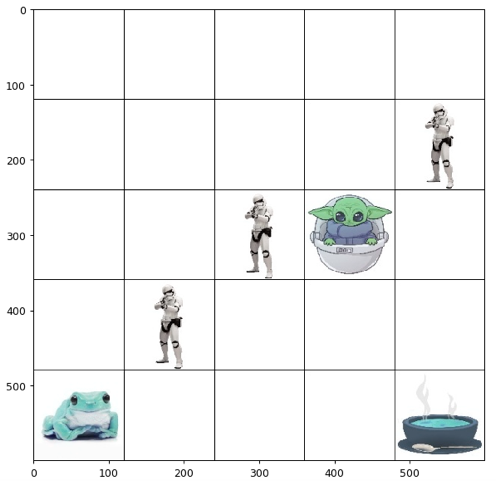
****

Figure 6f

Figure 6e

Figure 6d

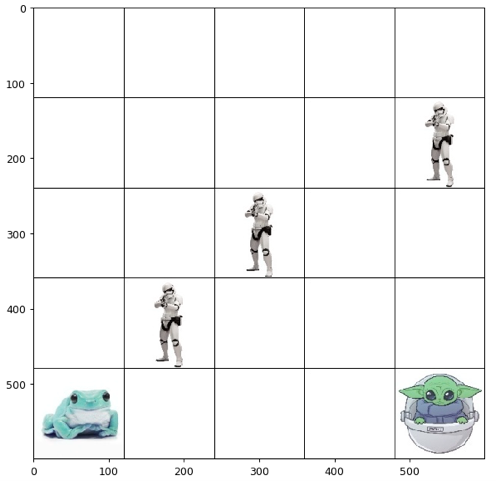
****

Figure 6g