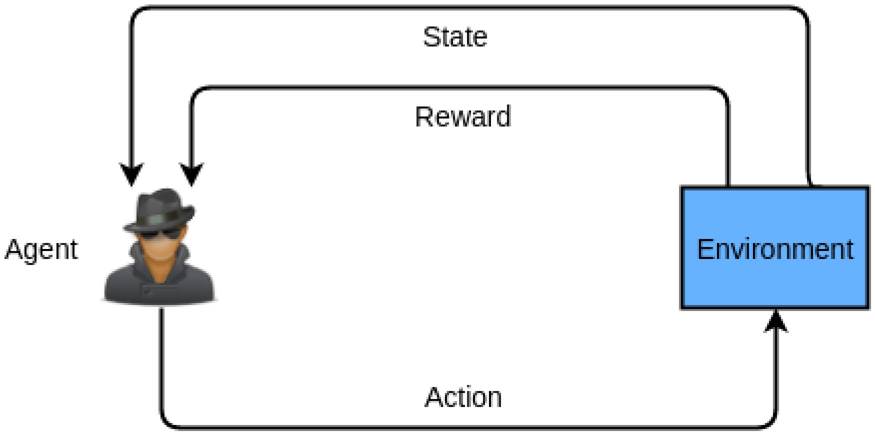
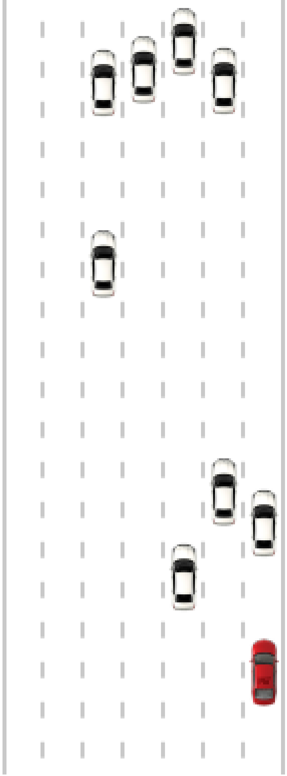
Reinforcement Learning
and
Deep Q-Network

Marax AI Inc

What is Reinforcement Learning?



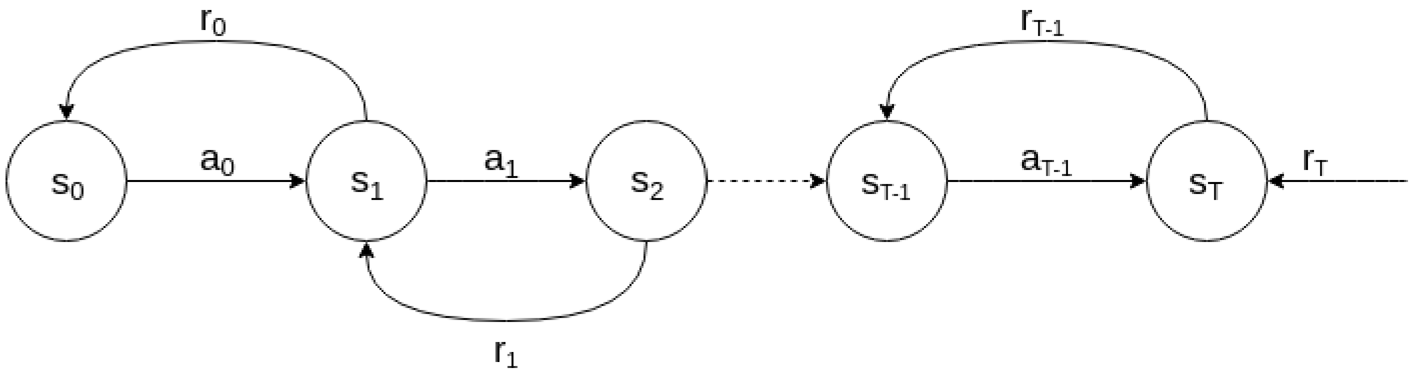


Learn through experience to find the sequence of actions to maximize the reward

Why?

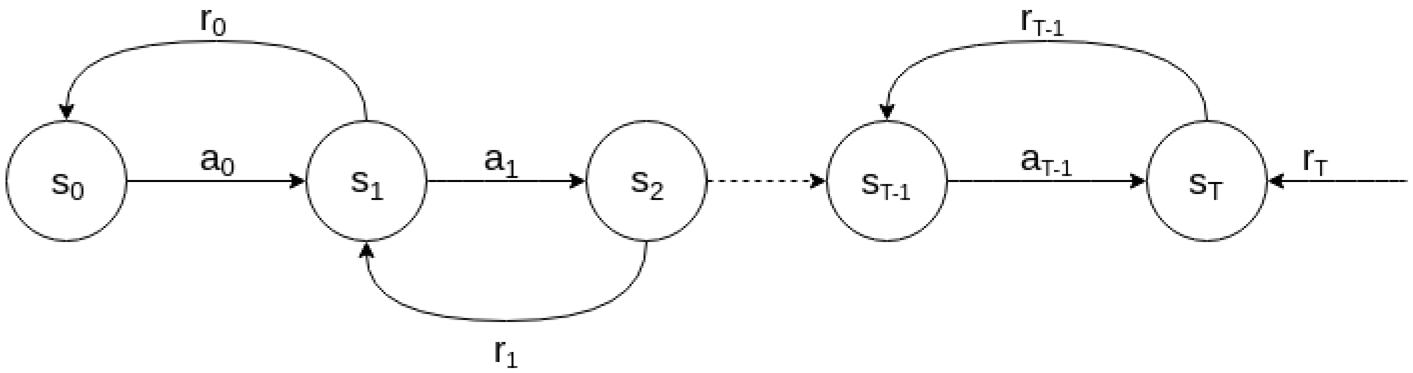
- Analogous to human learning
- Applications in robotics, healthcare, driving cars, consumer modelling etc.

Modelling the Problem



Find a policy $\pi(s) \Rightarrow$ sequence of actions, which maximizes the total rewards, ($r_0 + r_1 + \ldots + r_T$).

Modelling the Problem



Find a policy $\pi(s) \Rightarrow$ sequence of actions, which maximizes the total discounted rewards, ($r_0 + \lambda r_1 + \lambda^2 r_2 + \ldots + \lambda^T r_T$) - discounted for choosing early reward actions.

Q-value

At a given time t, given a state, s, and action, a, what would be the expected future reward.

$$Q(s_t, a_t) = E[R_t]$$

$$Q(s_t, a_t) = E[\sum_{i=t}^{T} \gamma^{i-t} r_i]$$

Q^*-value

At a given time t, given a state, s, and action, a, what would be the maximum expected future reward given an optimal policy π^*.

$$Q^*(s_t, a_t) = \max_{\pi} E[\sum_{i=t}^{T} \gamma^{i-t} r_i]$$

Bellman Equation

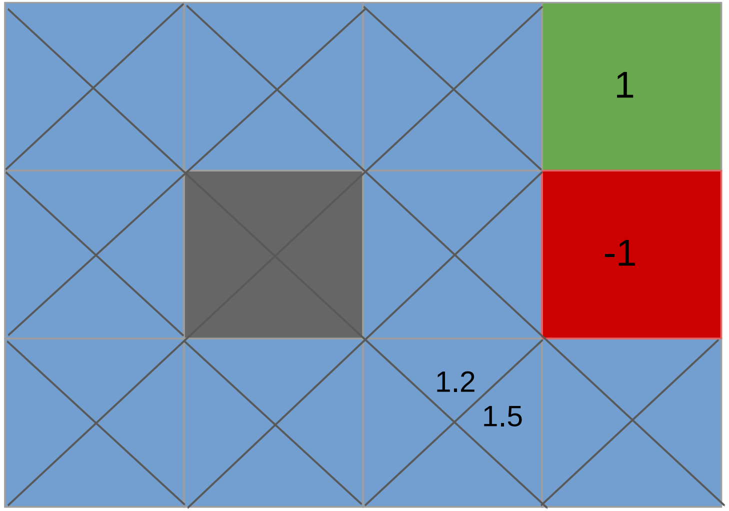
If we know the optimal value $Q^*(s_{t+1}, a_{t+1})$ at the next time step for all actions a_{t+1}, then the optimal Q-value at the current time step is given by,

$$Q^*(s_t, a_t) = E[r_t + \gamma Q^*(s_{t+1}, a_{t+1})]$$

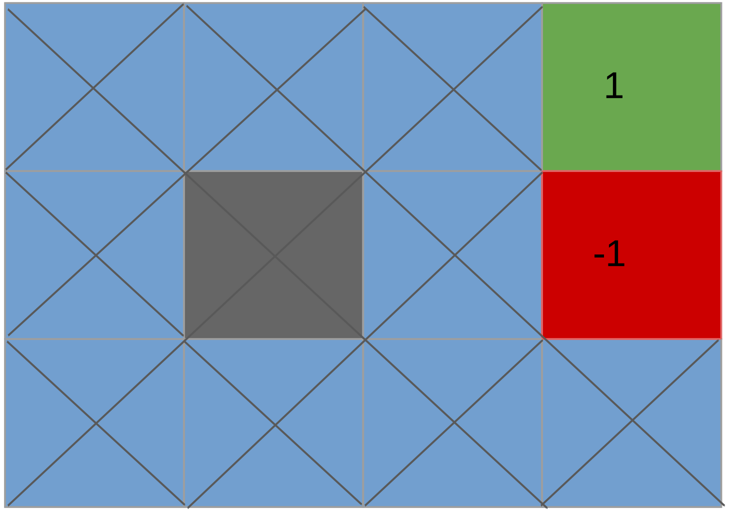
$$Q^*(s_{t+1}, a_{t+1})?$$

Exploration and Exploitation

At a given state s_t, if we decide to take action a_t, it does so with probability 1 - ε. With probability ε, our agent might land up in other states → ε-greedy exploration

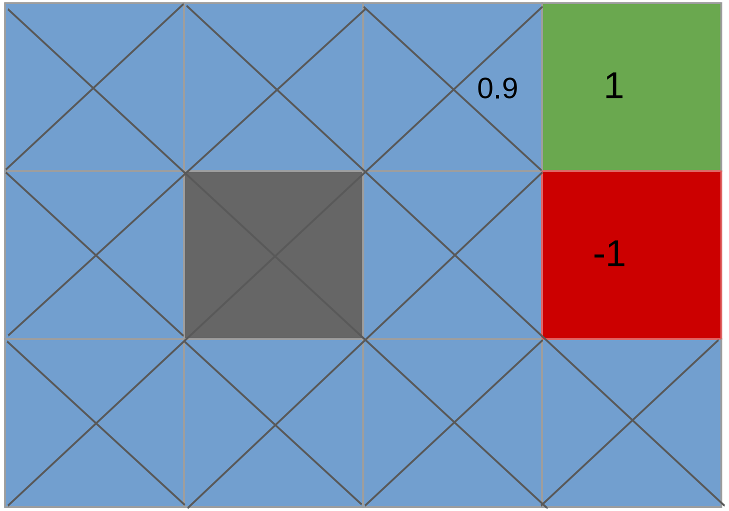


We initialize the q-values with some random values and then take actions for which q-value is maximum.



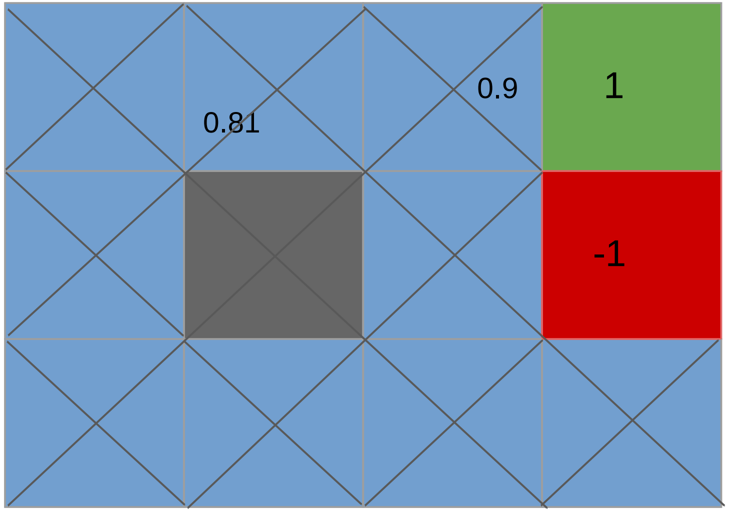
As number of iterations, $i \to \infty$, $Q \to Q^*$

We initialize the q-values with some random values and then take actions for which q-value is maximum.



As number of iterations, $i \rightarrow \infty$, $Q \rightarrow Q^*$

We initialize the q-values with some random values and then take actions for which q-value is maximum.



As number of iterations, $i \rightarrow \infty$, $Q \rightarrow Q^*$

Code Walkthrough - Quickly

Drawback

If there are large number of states and actions, often, the case in the real world scenario, it would take forever to find the optimal q-values for all the states.

Total number of (states,action) =

- Image - 84 x 84
- Gray-level - 256
- Temporal Window - 4

$$256^{84x84x4}$$

Deep Reinforcement Learning

Use neural network to approximate the optimal Q-value parameterized by weights, θ.

$$Q(s, a; \theta) \approx Q^*(s, a)$$

Deep Q-Network

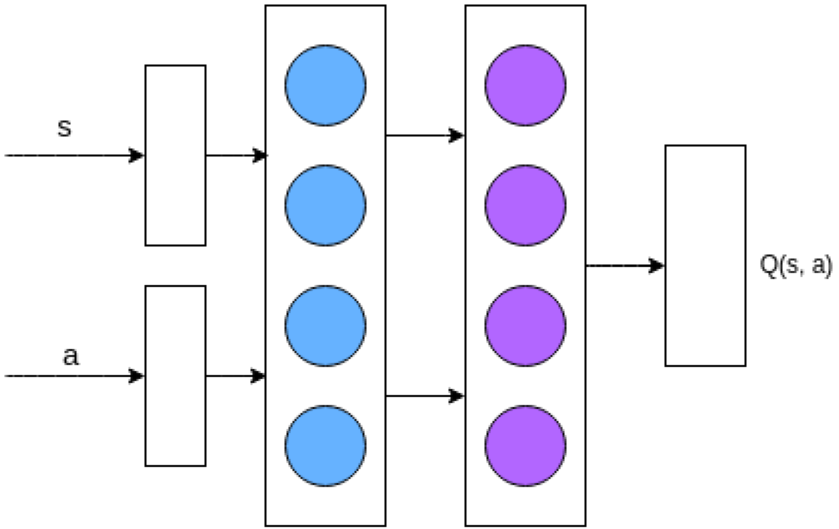
Input: State, s and Actions, a

Output: Q-value

Loss Function: ?

$$L_i(\theta) = E[(y_i - Q(s, a; \theta))^2]$$

where, $y_i = E[r + \gamma \max_{a'} Q(s', a'; \theta)]$



Algorithm

1. Initialize the weights with random values.
2. Do a forward pass with current state s and actions a, and calculate the Q(s,a).
3. Choose action based on epsilon greedy exploration and get next state s'.
4. Do a forward pass with next state s' and actions a' and find the maximum Q value.
5. Assign y_i(target) as the sum of reward received from step 2 and discounted Q-value calculated from the step 3 as depicted from the equation,

$$y_i = r + \gamma \max_{a'} Q(s', a'; \theta)$$

6. Calculate the loss from the result received from step 2 and 4.
7. Use back-propagation to update the weights.

Demo

- http://selfdrivingcars.mit.edu/deeptrafficjs/
- Trained Model Video

Challenges

- Traffic Light Detection
- Lane Detection
- Pedestrian Detection
- Many More…

Tesla Video

Questions?

☺ Thank you ☺