Deep Q-Learning

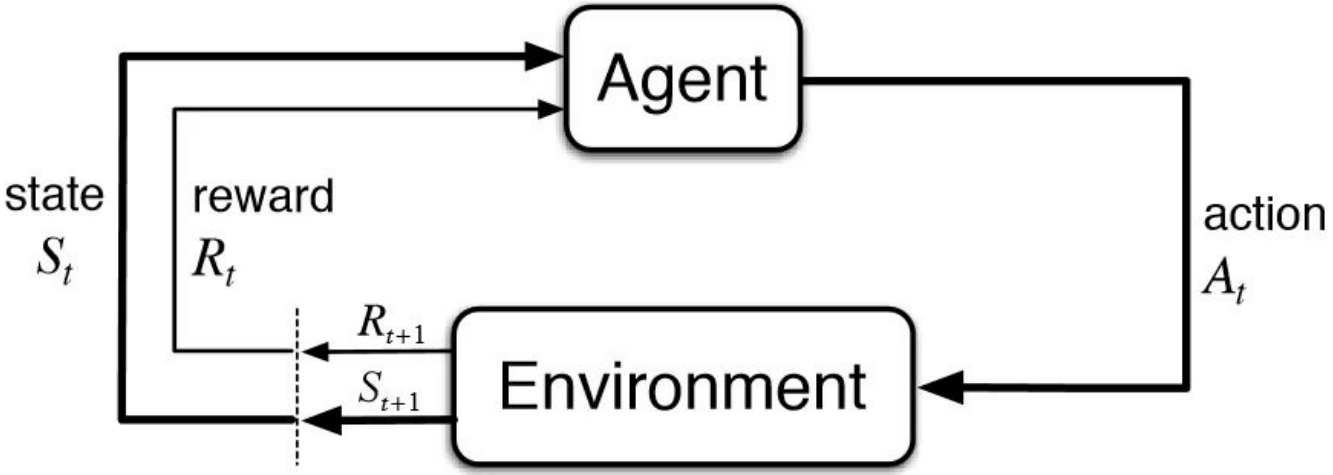
Presentation by Rahul Vishwakarma

Content

- Reinforcement Learning
- Q-Learning
- Epsilon greedy policy
- Bellman's Equation
- Deep Q-Learning
- CartPole Game

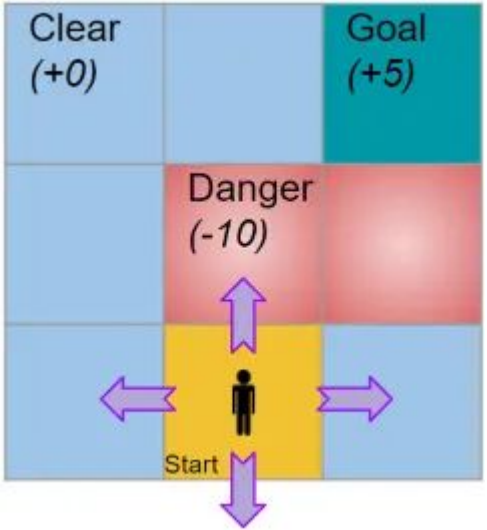
Reinforcement Learning

Training an agent to interact with an environment in order to maximize the cumulative reward over time.



Q-Learning

The Q-learning algorithm uses a Q-table of State-Action Values (also called Q-values). This Q-table has a row for each state and a column for each action. Each cell contains the estimated Q-value for the corresponding state-action pair.



	Left	Right	Up	Down
(1,1)	0	0	0	0
(1,2)	0	0	0	0
(1,3)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
(2,3)	0	0	0	0
(3,1)	0	0	0	0
(3,2)	0	0	0	0
(3,3)	0	0	0	0

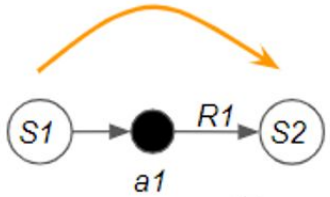
1 Initialise Q-Value (ie. State Action value) estimates with zero value, and pick the initial state.

	a1	a2	a3	a4
S1	0	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	0	0	0	0
S5	0	0	0	0

S1 + initial state

2 Agent picks an **action to execute** from the current state using ε-greedy policy.

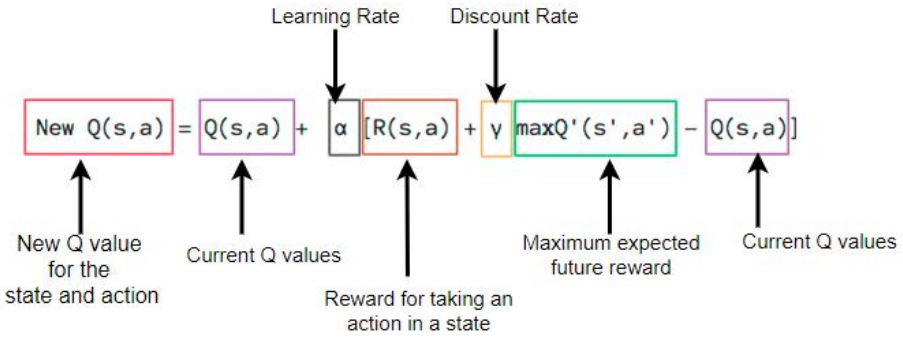
3 Agent obtains observation data from the environment *(S1, a1, R1, S2)*

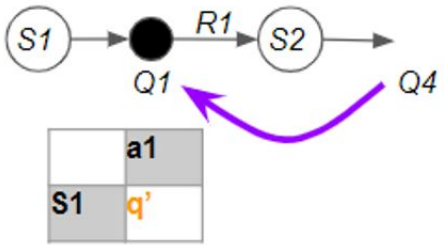


S2 Next state becomes the Current State

4 Update the **current Q-value** using the observed reward and the **target Q-value.** The target Q-value is the action with the max Q-value from the next state.

$$Q(S,A) = Q(S,A) + \alpha(R + \gamma\, max\, Q(S',a) - Q(S,A))$$





State

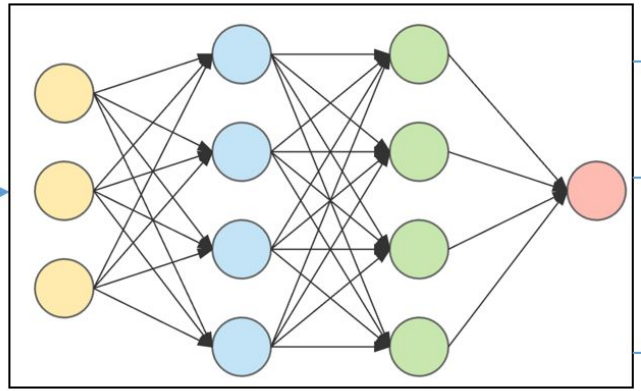
Action

Q Table	
State-Action	Value
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0

Q-Value

Q Learning

State



Q-Value Action 1

Q-Value Action 2

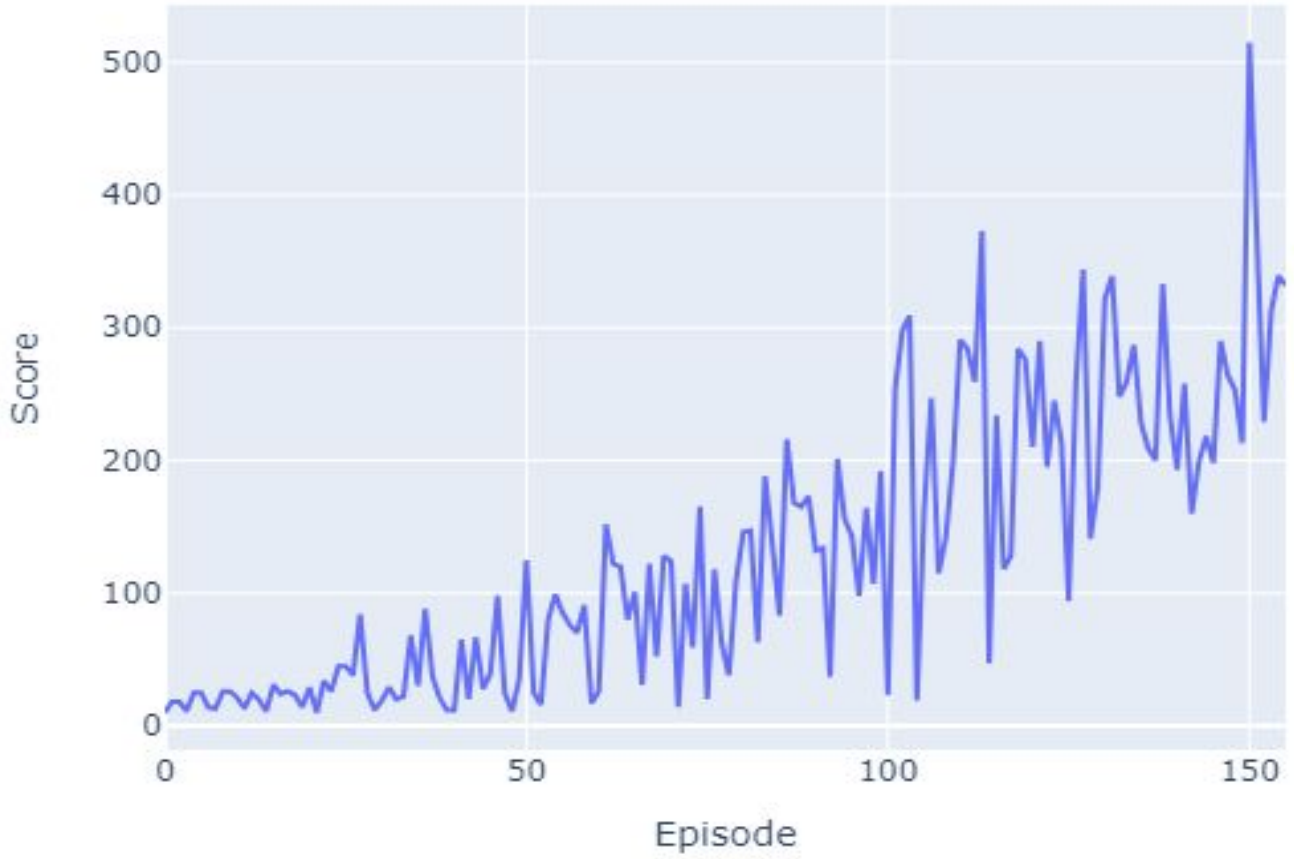
-

-

Q-Value Action N

Deep Q Learning

Continued from here on Jupyter notebook with the CartPole Game


Score
500
400
300
200
100
0
0
50
100
150
Episode

Thank You