Deep Q-Learning

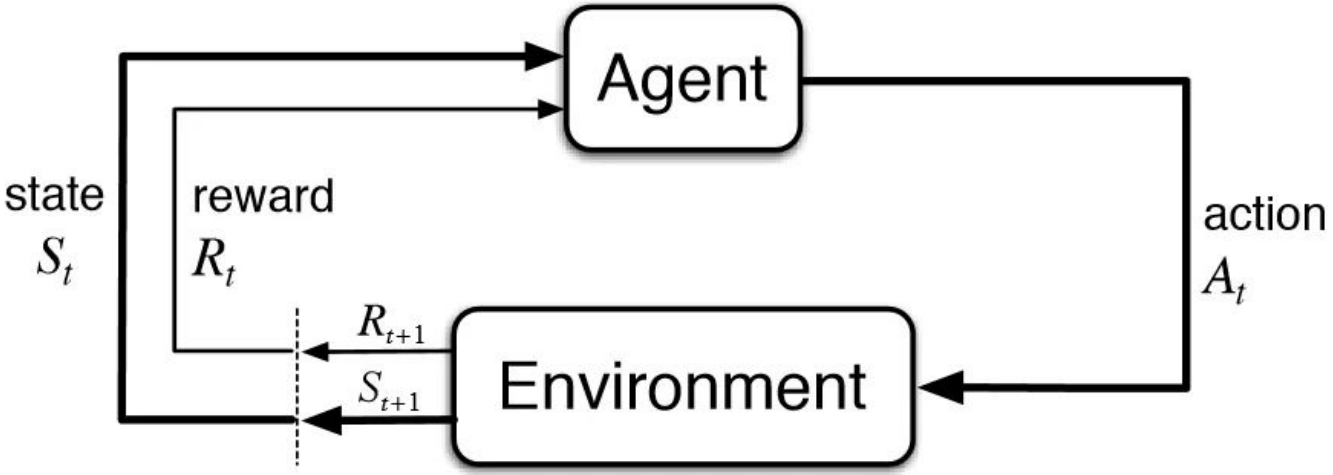
Presentation by Rahul Vishwakarma

Content

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

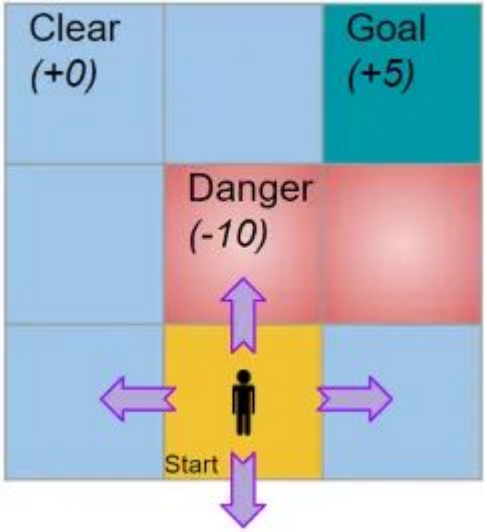
Reinforcement Learning

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.
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.

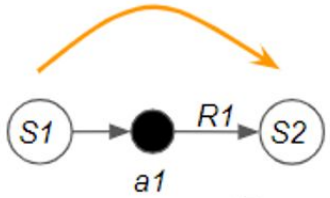
	a1	a2	a3	a4
S1	0	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	0	0	0	0
S5	0	0	0	0

S1

+ initial state

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

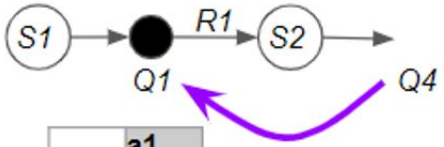
3 Agent obtains observation data from the environment *(S1, a1, R1, S2)*



S2 Next state becomes the Current State

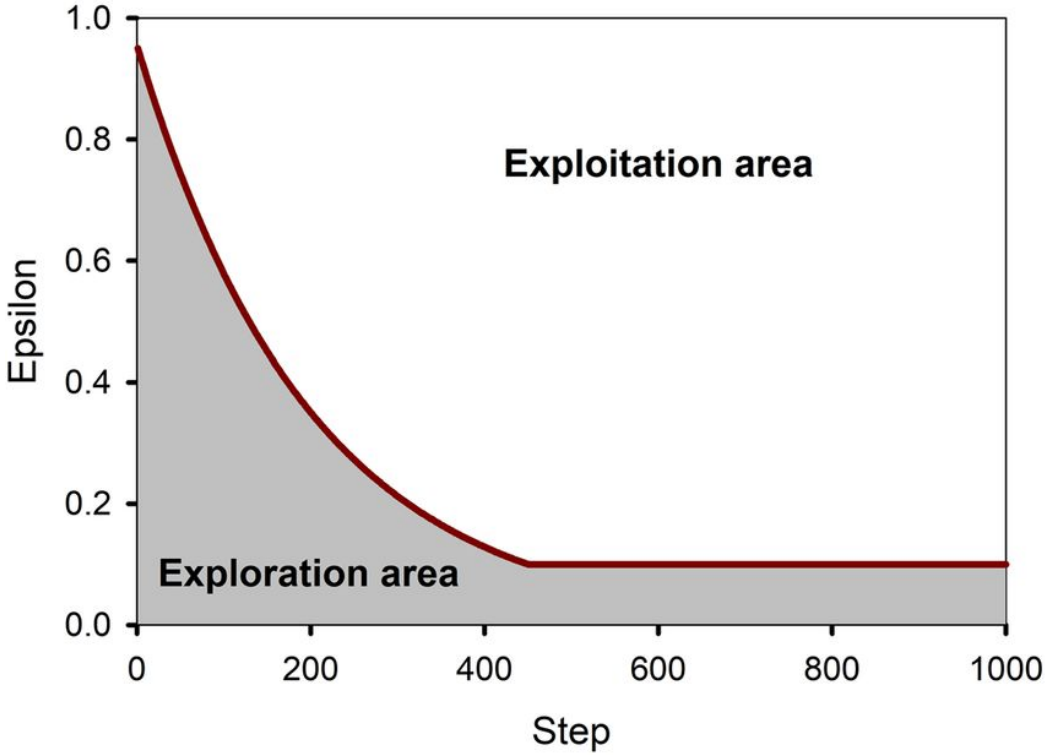
4 Update the **current Q-value** using the observed reward and the **target Q-value.** The target Q-value is the action with the max Q-value from the next state.

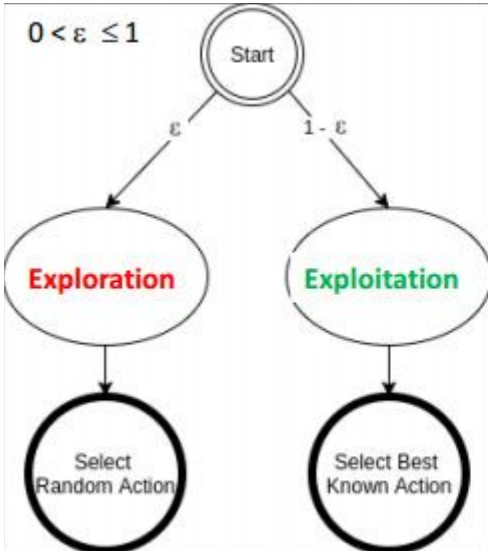
$$Q(S,A) = Q(S,A) + \alpha(R + \gamma\, max\, Q(S',a) - Q(S,A))$$



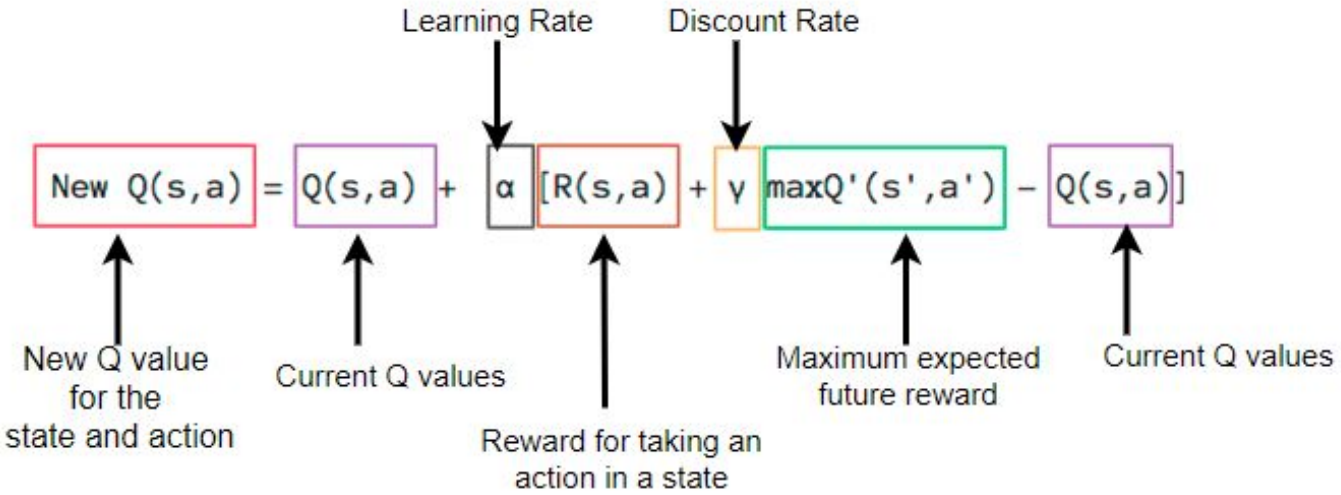
	a1
S1	q'

Epsilon Greedy Policy





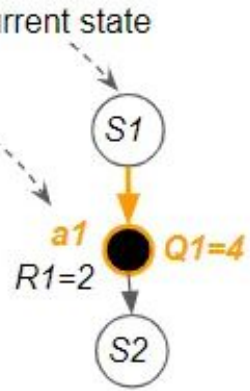
Bellman's Equation

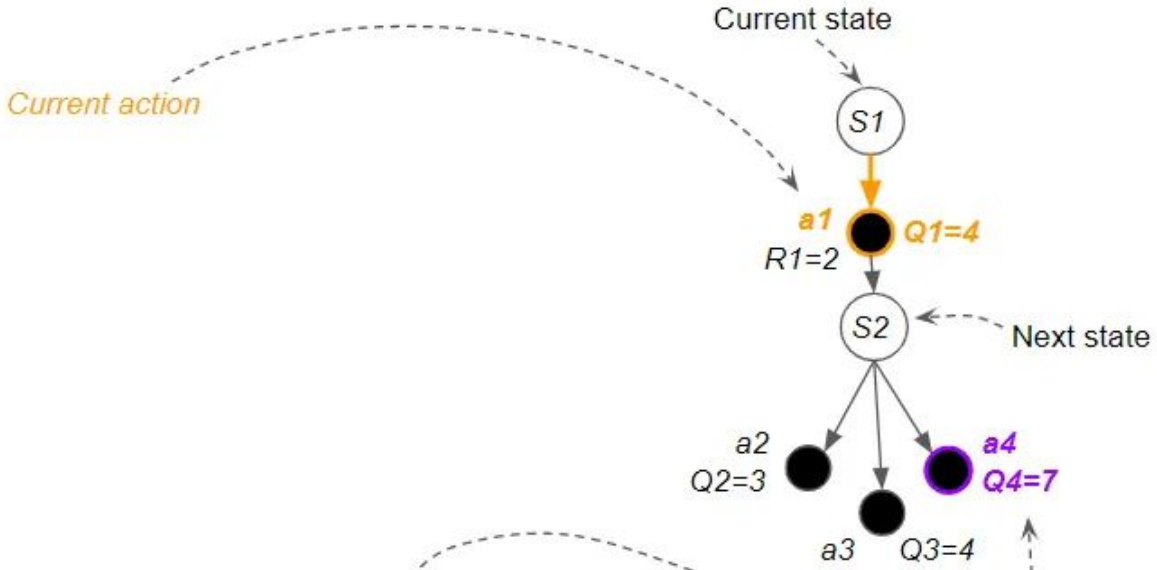


2

Current action picked using an ε-greedy policy.

	a1	a2	a3	a4
S1	4	9	2	3
S2	0	3	4	7

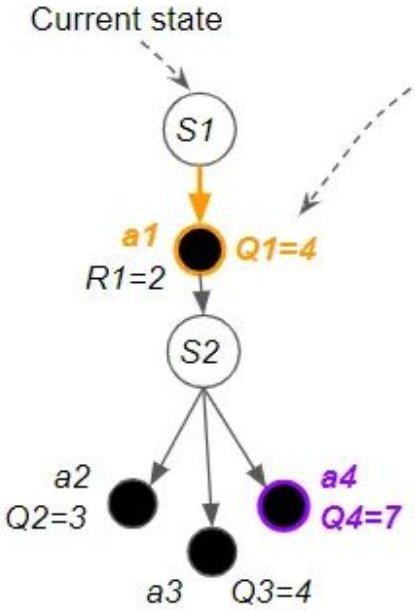




4a

The target action is the action from the next state with the highest Q-value

	a1	a2	a3	a4
S1	4	9	2	3
S2	0	3	4	7



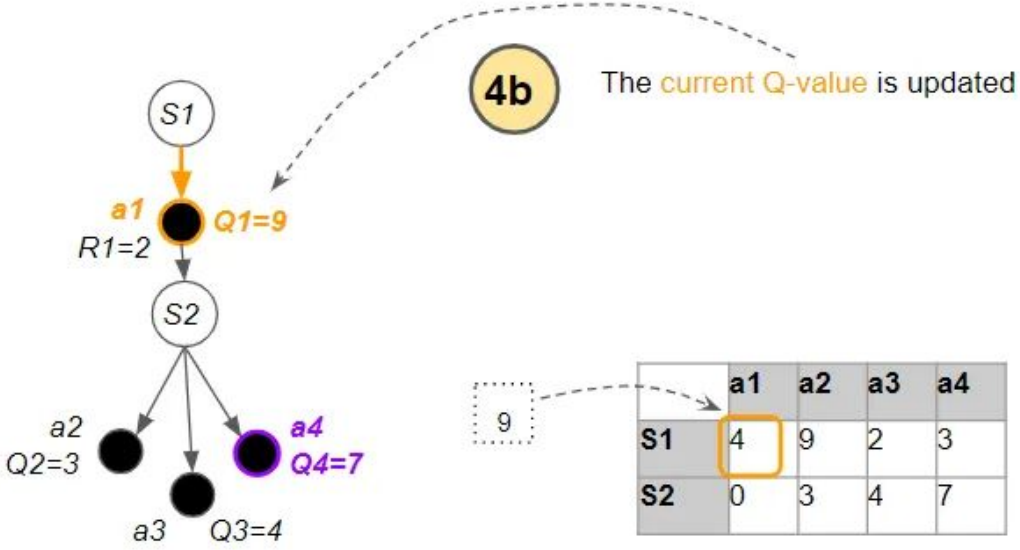
4b The current Q-value is updated

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

Q1 = Q1 + α (R1 + γ * Q4 - Q1)

Q1 = 4 + (2 + 7 - 4) = 9

NB: taking α = γ = 1 for simplicity


4b
The current Q-value is updated
S1
a1
Q1=9
R1=2
S2
a2
Q2=3
a4
Q4=7
a3
Q3=4
9
a1
a2
a3
a4
S1
4
9
2
3
S2
0
3
4
7

State

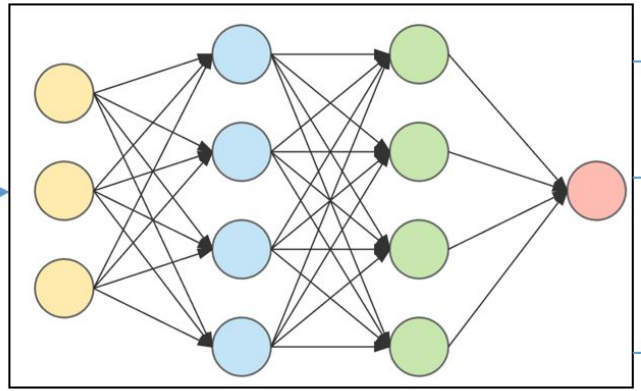
Action

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

Q Table vs Neural Network (DQN)

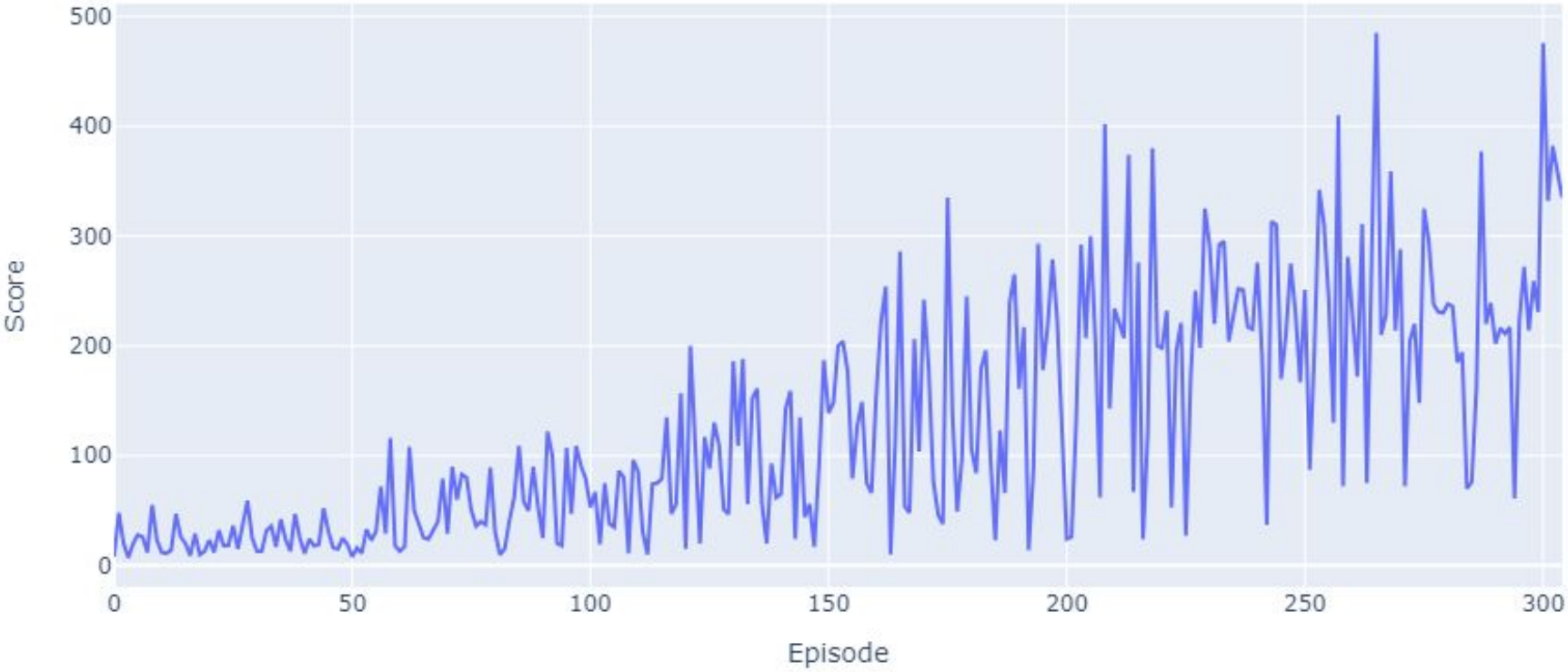
Scalability: Q-tables become impractical as the number of states and actions in the environment grows, making it infeasible to store and update all Q-values explicitly.

Continuous action spaces: In environments with continuous action spaces, it's challenging to create a Q-table that covers all possible actions finely.

Memory requirements: Neural networks typically require less memory than Q-tables when dealing with large state spaces because they store function approximations rather than explicit values for every state-action pair.

Function approximation: Q-tables can only represent linear relationships and are limited in their modeling capacity.

CartPole Game in Jupyter Notebook


Score
500
400
300
200
100
0
0
50
100
150
200
250
300
Episode

References

- Towards Data Science: Reinforcement Learning Explained Visually (Part 4): Q Learning, step-by-step
- Analytics Vidhya: A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python
- OpenAI Gymnasium Documentation: https://gymnasium.farama.org/environments/classic_control/cart_pole/

Thank You