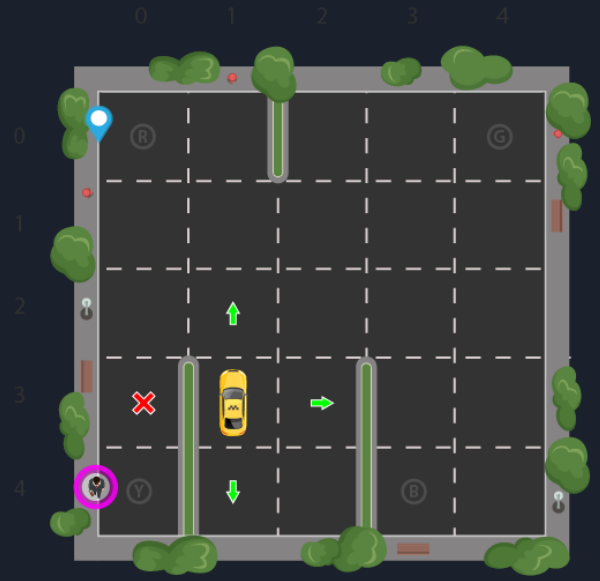
Taxi Driver Agent
Reinforcement Learning

Panagiotis Giannakopoulos

Problem Description

The taxi agent should pick up the passenger from one destination and drop him off to another without significant delays and following safety and traffic rules .

The solution is given by reinforcement learning algorithms.

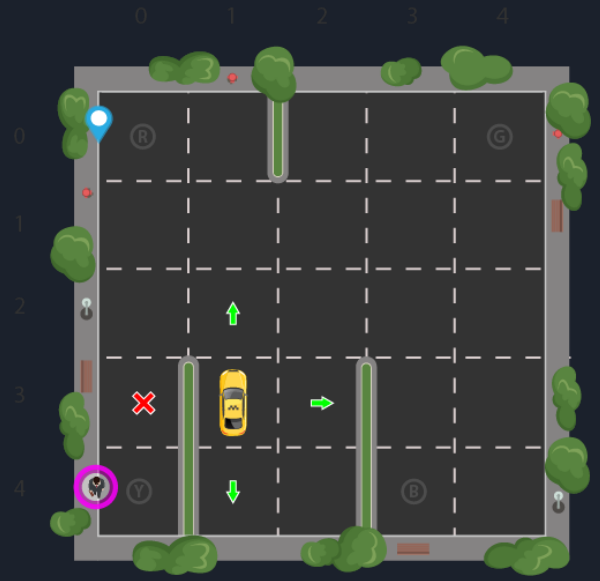


The problem is solved using the Open AI gym Taxi v3 simulator and Python3 libraries

Characteristics

Rewards

1. High positive reward for a successful drop-off because this behavior is highly desired.
2. Penalty for drop off the passenger in wrong location.
3. Slight negative reward at every time-step for not making it to the destination.

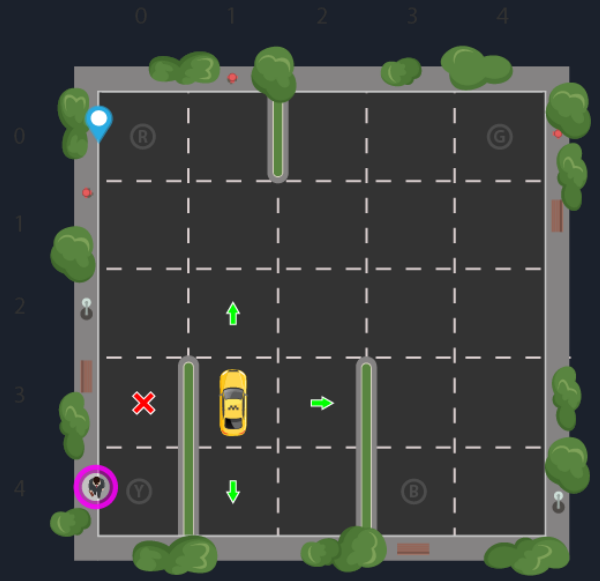


Characteristics

State Space

1. 5x5 grid of the parking lot ➔ 25 possible taxi locations.
2. 4 locations that we can pick up and drop off a passenger.
3. The passenger could be in one of the 4 locations (R, G, Y, B) or inside the taxi. ➔ 5 possible locations of the passenger.

Total number of possible states is 25×4×5 = 500

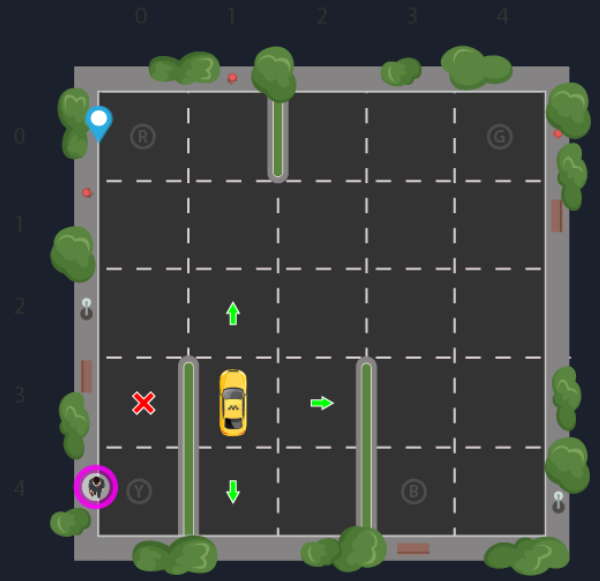


Characteristics

Action Space

1. south
2. north
3. east
4. west
5. pickup
6. drop-off

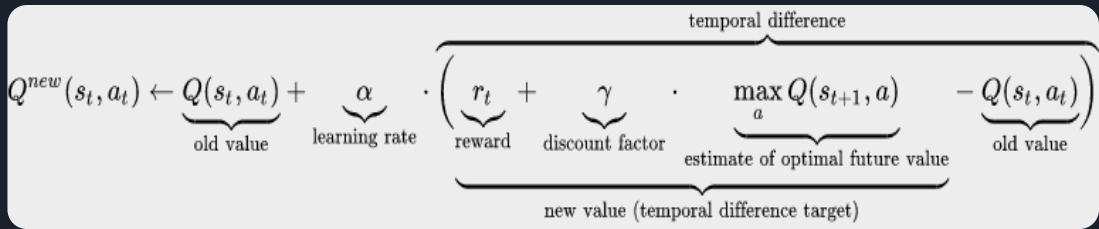
Total number of possible actions is 6



Reinforcement Learning Algorithms

Q Learning

- The goal to learn what action to take under what circumstances.
- Q table is initialized to a arbitrary fixed value.
- Each time t the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} and then Q is updated.
- The core of the algorithm is a simple value iteration update, using the weighted average of the old value and the new information



Reinforcement Learning Algorithms

SARSA Learning

- SARSA algorithm is similar to Q learning.
- The agent interacts with the environment and updates the policy based on actions taken.
- The Q value for a state-action is updated by an error, adjusted by the learning rate a.
- Q values represent the possible reward received in the next time step for taking action a_t in state s_t, plus the discounted future reward received from the next state-action observation.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Reinforcement Learning Algorithms

Deep Q Network

- Neural networks are function approximators and can learn to map states to values, or state-action pairs to Q values.
- At the beginning of reinforcement learning, the neural network coefficients may be initialized randomly.
- Using feedback from the environment, the network can use the difference between its expected reward and the ground-truth reward to adjust its weights and improve its interpretation of state-action pairs.

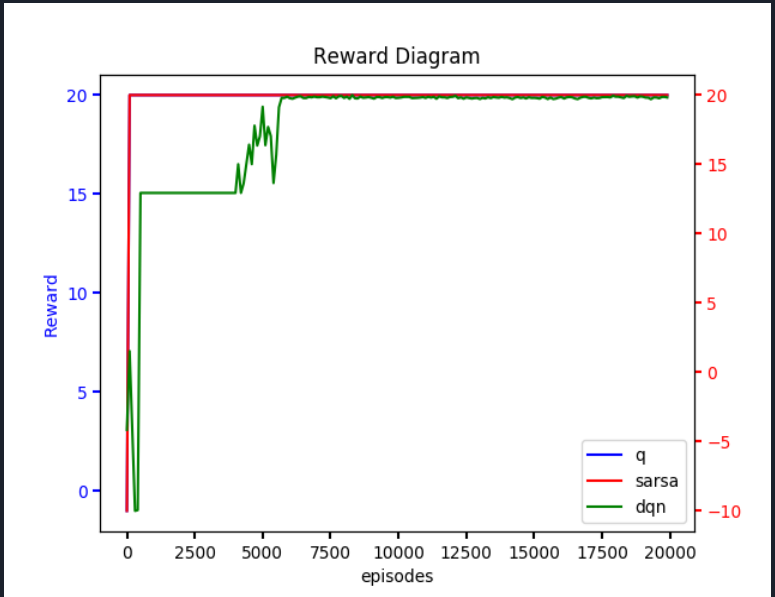
Hyperparameter Search

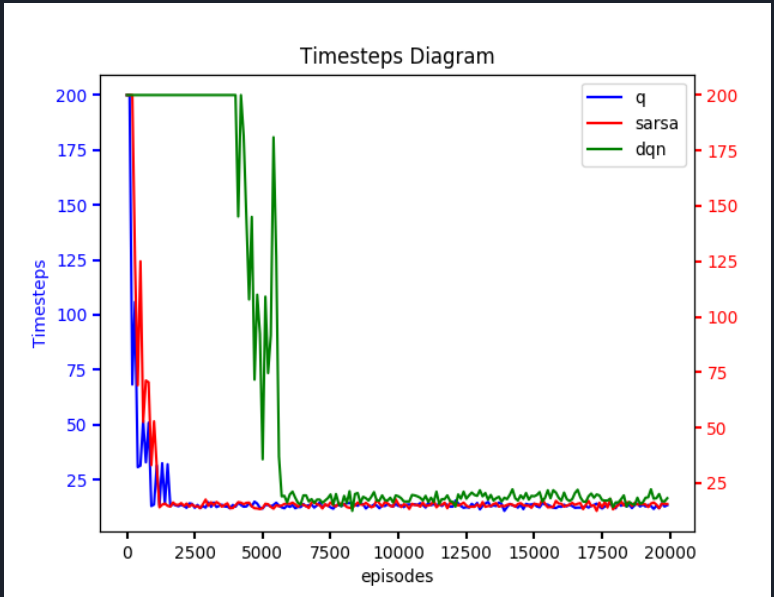
- Grid search is applied on a limited range of the hyperparameters in order to improve the performance of Q and SARSA learning.
- The search function selects the parameters that would result in the best reward/timesteps ratio.
- The selection of this ratio gives parameters which enable us to get the maximum reward as fast as possible.

	Learning rate α	Discount factor γ	Exploration/Exploitation parameter ε
Q	0.75	0.9	0.9
SARSA	0.5	0.975	0.9

Results

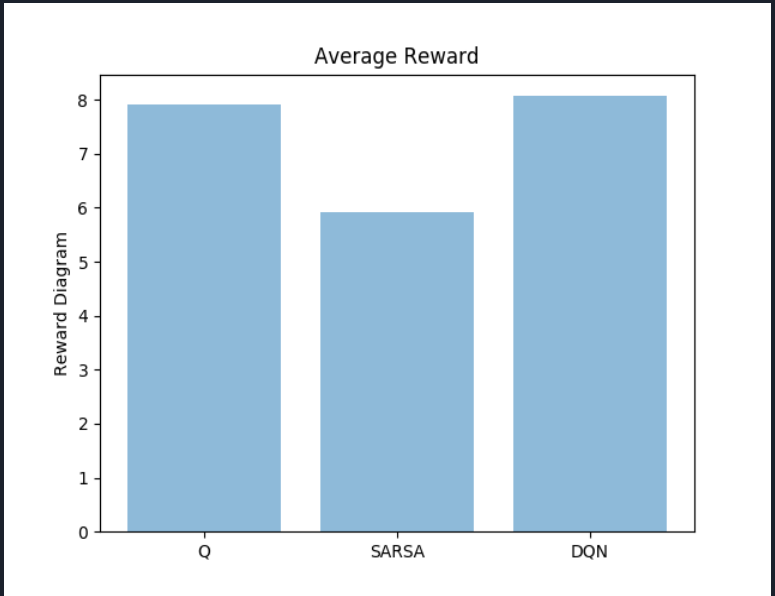
Performance of each algorithm during the training process per episode.

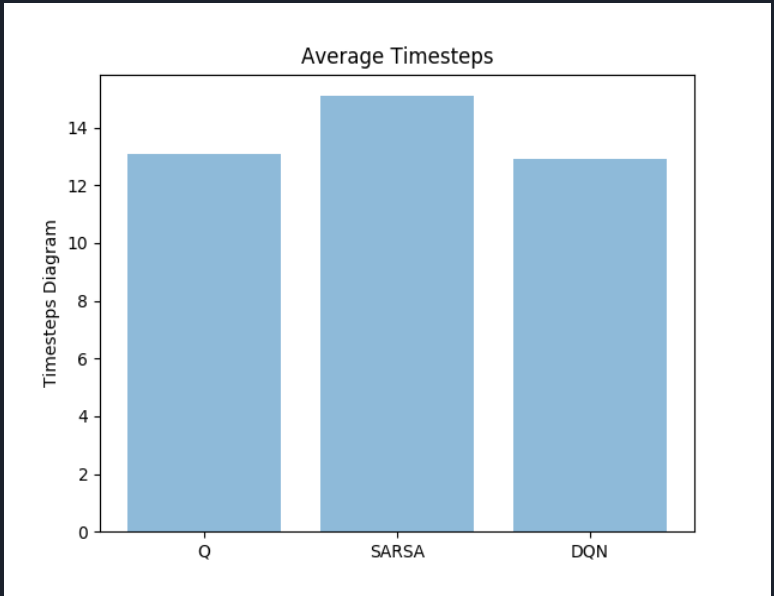




Results

Average output of each algorithm for 20.000 episodes after training.





Conclusions

1. Each algorithm goals to minimize the total timesteps and the same time to maximize the total reward of the episode.
2. During the training process, Q and SARSA learning perform similarly while DQN converges more slowly.
3. At the end of the training process, the average behavior of Q and DQN is similar while SARSA has lower performance .
4. All algorithms give us good enough results.
5. Q learning has the best performance during the training process and at the final simulation.

Thank you!