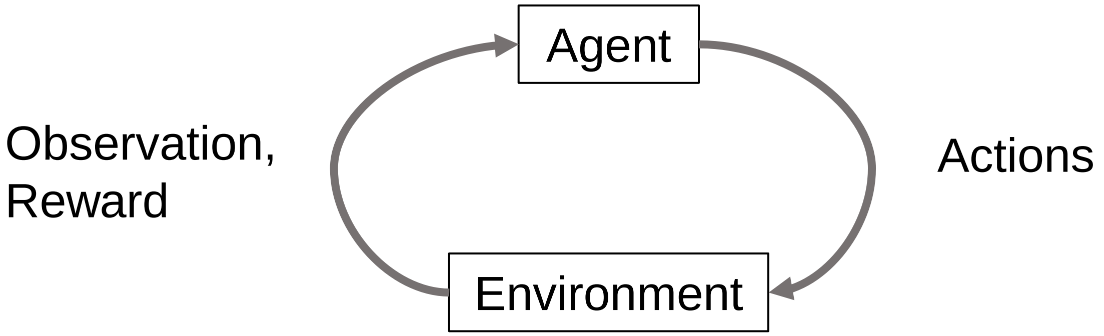
Q-learning vs Double Q-learning

Kristian Angeli Pajanonot

Reinforcement Learning

Overview



Reinforcement Learning

One large challenge in reinforcement learning is the exploration-exploitation dilemma.

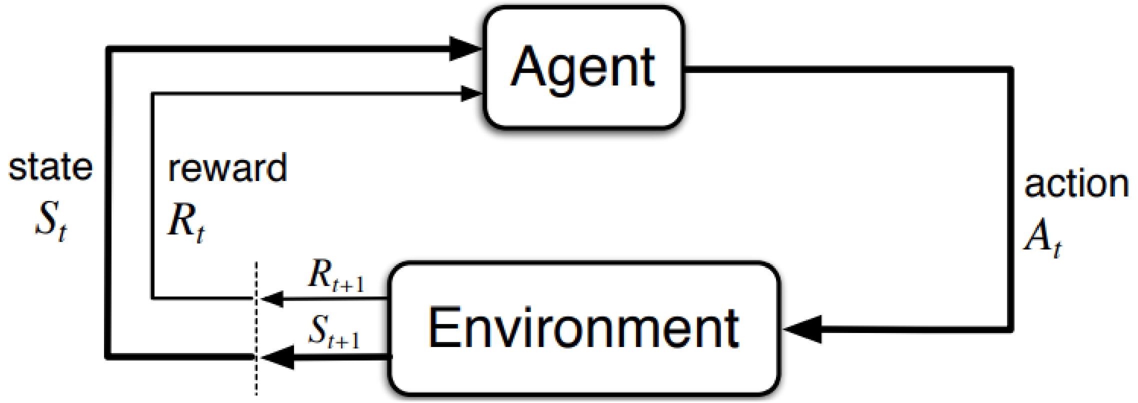
- The agent should **exploit** known actions in order to maximize its total reward
- The agent should **explore** unknown actions in order to discover actions that are more rewarding than the ones it already knows

ε-greedy method – to balance between exploration and exploitation

Reinforcement Learning

Markov Decision Process

Many reinforcement learning problems can be mathematically formalized as finite Markov decision processes



The agent–environment interaction in a Markov decision process

Consider episodic problems: During an episode, the agent tries to maximize the total expected discounted return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \ldots \tag{1}$$

γ is the discount factor that determines the importance of future rewards

Image Source: Sutton and Barto, 2018

Reinforcement Learning

Markov Decision Process

A policy π is a mapping from states to probabilities of selecting each possible action.

To find the optimal policy given only the experience $(S_t, A_t, R_{t+1}, S_{t+1}, \ldots)$ of an environment, we consider the action-value function for policy π: $Q_\pi(s, a)$

$$Q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] \tag{2}$$

Once one finds the optimal value $Q^*(s, a)$:

$$Q^*(s, a) = E[R_{t+1} + \gamma\, max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a] \tag{3}$$

The optimal policy is easily found just by taking the action that maximizes it.

$$\pi^*(a, s) = argmax_a Q^*(s, a) \tag{4}$$

Q-learning

Algorithm

In Q-learning, the learned action-value function Q, directly approximates Q^*, the optimal action-value function, independent of the policy being followed. The update rule for an approximation Q for a sampled trajectory $S_t, A_t, R_{t+1}, S_{t+1}$ is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

α is the step size

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha \big[R + \gamma \max_a Q(S', a) - Q(S, A) \big]$
    $S \leftarrow S'$
  until S is terminal

Image Source: Sutton and Barto, 2018

Maximization (Overestimation) Bias

Problem with Q-learning

The overestimation bias occurs since the $max_a \, Q(S_{t+1}, a)$ is used in the Q-learning update.

In Q-learning, the maximum over estimated values is used as an estimate of the maximum of the true values. To see why:

Consider the case that there is some source of random approximation error, such as stochastic rewards. Therefore, for all actions a, we have:

$$Q(S_{t+1}, a) = \underset{\substack{\| \\ 0}}{Q^*(S_{t+1}, a)} + \; e(S_{t+1}, a) \tag{5}$$

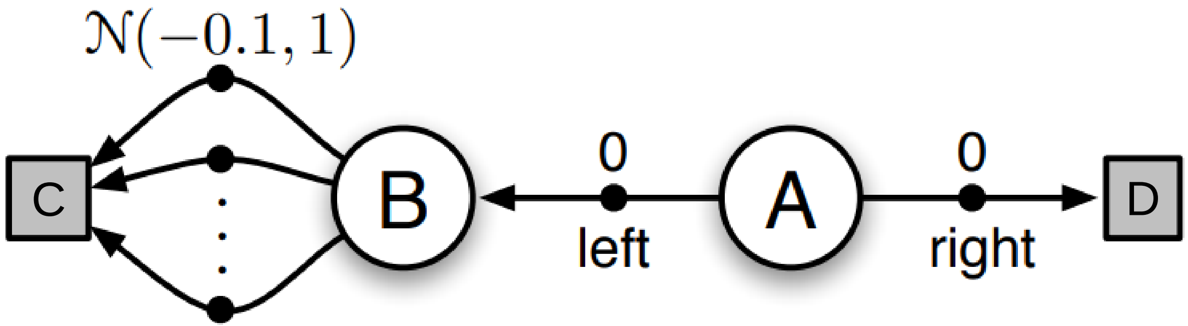
$e(S_{t+1}, a)$ is a positive or negative noise term

The estimated values $Q(S_{t+1}, a)$ are uncertain and distributed some above and some below zero

The maximum of the true values is zero, but the maximum of the estimates is positive, a positive bias – maximization bias

Maximization Bias: Example

Problem with Q-learning



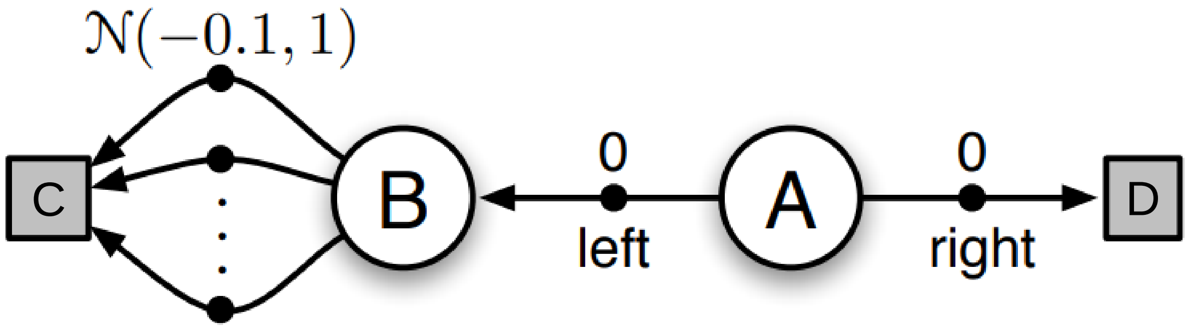
An episodic finite Markov decision process to highlight the problems caused by overestimation bias

- A and B are non-terminal states. A is the starting state. C and D are terminal states.
- The agent can take two actions: **left** and **right**
- Reward from taking the actions:
  - A to B = 0
  - A to D = 0
  - B to C = normal distribution with mean -0.1 and variance 1.0

Example from Sutton and Barto, 2018

Maximization Bias: Example

Problem with Q-learning



An episodic finite Markov decision process to highlight the problems caused by overestimation bias

- Expected return for any trajectory starting with **left** is -0.1 (with $\gamma = 1$). Expected return for any trajectory starting with **right** is 0
- Taking **left** from A is a bad idea. Therefore, the **optimal policy is to choose right** from A.
- However, a Q-learning agent following an ε-greedy policy could choose left many times in the beginning of learning, because it overestimates the maximum optimal action value of B (because some of the values of the reward are positive, the agent will be *tricked* to consider that taking action left from A maximizes the reward)

Example from Sutton and Barto, 2018

Double Q-learning

Algorithm to avoid maximization bias

In Q-learning, there will be a maximization bias if we use the maximum of the estimates as an estimate of the maximum of the true values.

$$max_a\, Q(S_{t+1}, a) = Q(S_{t+1}, \arg max_a\, Q(S_{t+1}, a))$$

Use same values to select the maximizing action, and to estimate its value

Instead, decouple selection and evaluation, and have two action-value functions: $Q_1(S_{t+1}, a_1), Q_2(S_{t+1}, a_2)$

- Use one estimate Q_1 to determine the max action $a^* = argmax_a Q_1(S_{t+1}, a)$
- Use the other estimate Q_2 to estimate the value of a^*: $Q_2(S_{t+1}, a^*) = Q_2\big(S_{t+1},\ argmax_a Q_1(a)\big)$
- Since Q_2 was updated on the same problem, but with a different set of experience samples, this can be considered an unbiased estimate for the value of this action: $E\big(Q_2(S_{t+1}, a^*)\big) = Q(S_{t+1}, a^*)$
- We can also repeat the process with the role of the two estimates reversed to yield a second unbiased estimate $Q_1\big(S_{t+1},\ argmax_a Q_2(a)\big)$. Update each estimate with 0.5 probability.

Double Q-learning

Algorithm

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

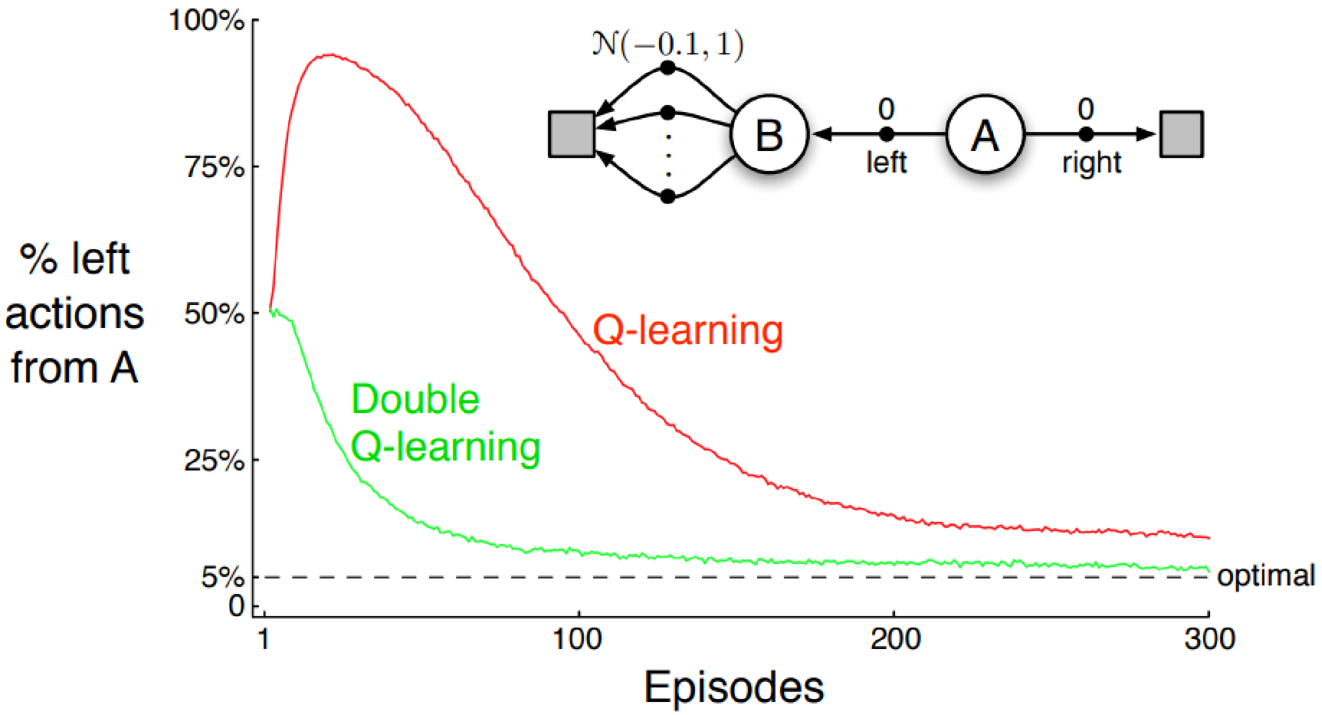
$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

    $S \leftarrow S'$
  until S is terminal

Image Source: Sutton and Barto, 2018

Q-learning vs Double Q-learning

Example



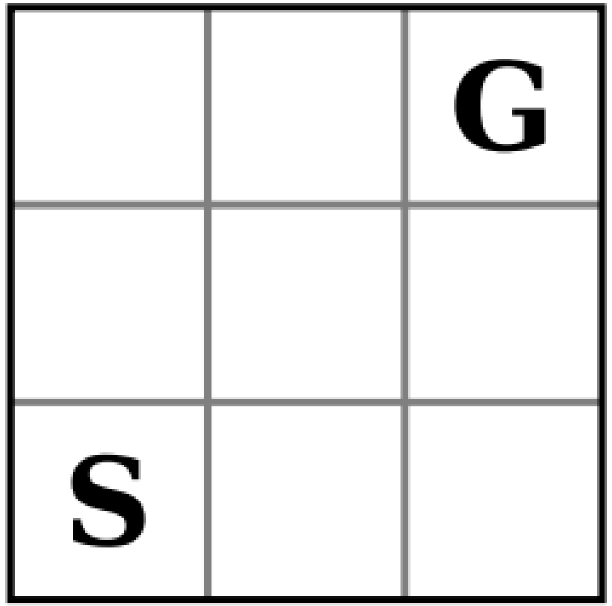
Comparison of Q-learning and Double Q-learning on a simple episodic MDP

Due to the maximization bias, Q-learning spends much more time selecting suboptimal actions than double Q-learning

Image Source: Sutton and Barto, 2018

Q-learning vs Double Q-learning

Grid World



Grid World

- Each state has 4 actions, corresponding to the directions the agent can go (agent can move up, down, left, right).
- The starting state (S) is in the lower left position and the goal state (G) is in the upper right.
- Each time the agent selects an action that walks off the grid, the agent stays in the same state.
- For each non-terminating step, the agent receives a random reward of −12 or +10 with equal probability.
- Once it arrives to a goal state it gets a reward +5 and ends an episode.

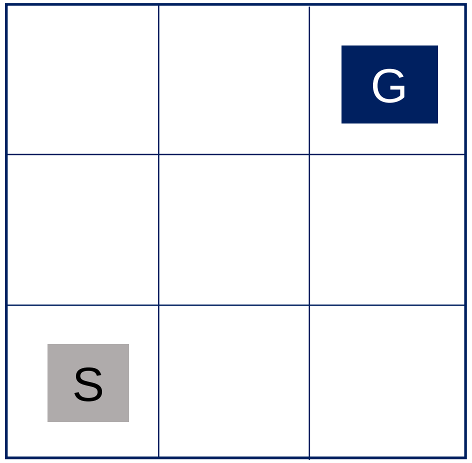
Image Source: van Hasselt, 2010

Optimal Policy, Reward

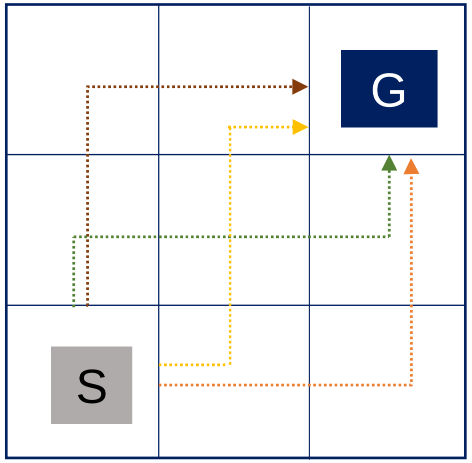
Grid World

The optimal policy ends an episode after 5 steps

The sum of average reward per state after an episode is +1



Grid World



Path the agent can take to go from S to G

-1	-1	+5
-1	-1	-1
-1	-1	-1

Average reward per state

Sum of rewards per episode (averaged over number of experiments)

Results: 500 episodes

Parameters:

$\gamma = 0.95$

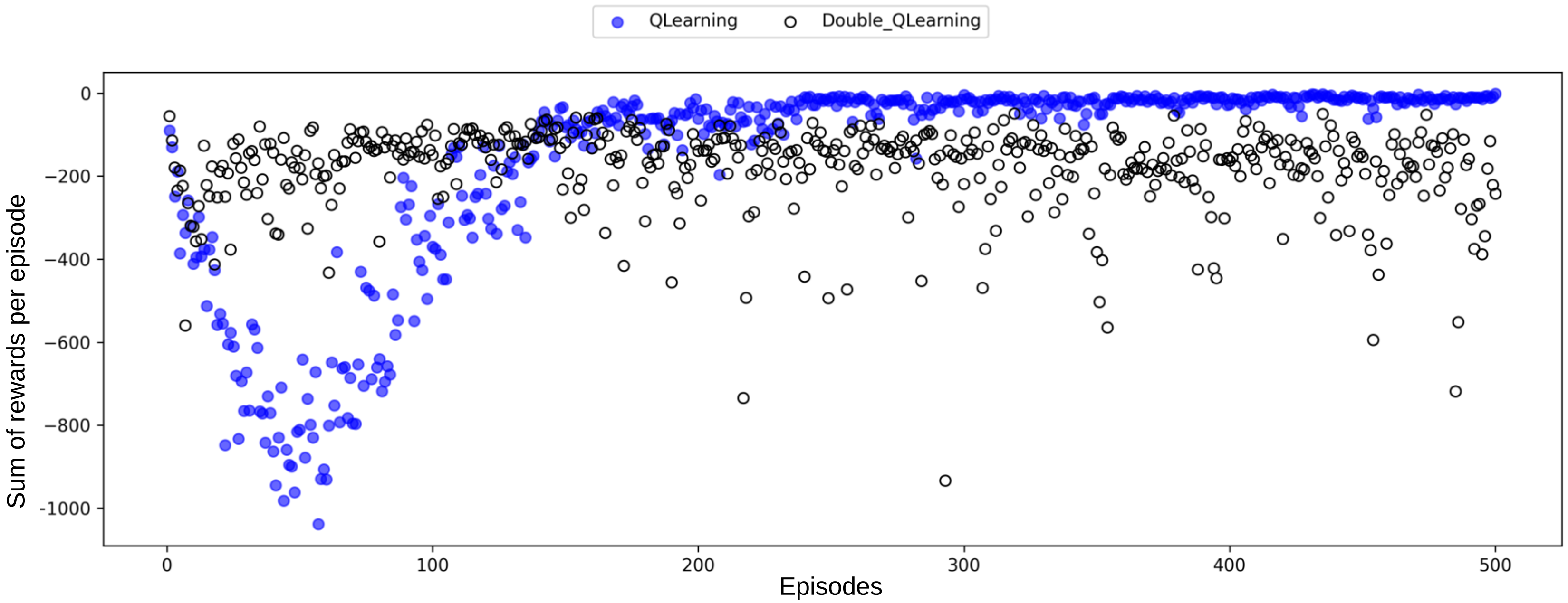
$number\ of\ trials = 200$

$$\varepsilon = \frac{1}{\sqrt{n(s)}}$$

$n(s) = number\ of\ times\ state\ s\ has\ been\ visited$

$$\alpha = \frac{1}{n(s,a)^{0.8}}$$

$n(s,a) = number\ of\ updates\ for\ each\ state, action$

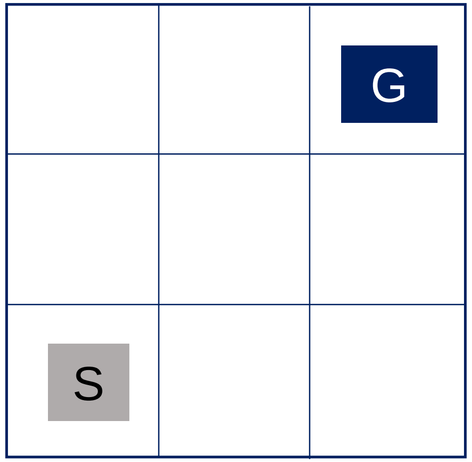


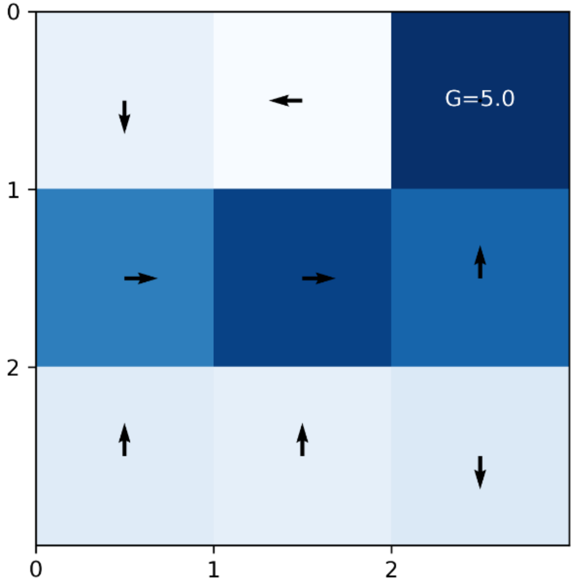
Result in grid world for Q-learning and Double Q-learning: Sum of rewards per episode (averaged over number of experiments)

Double Q-learning as measured in (average) sum of rewards per episode performs better than that of Q-learning in the first few episodes in this setting.

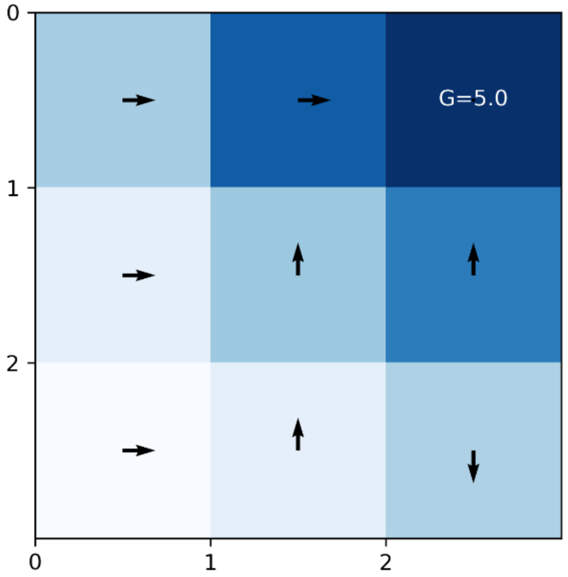
Snapshot of a Policy

Results: 500th episode





Q-learning



Double Q-learning

Action from the starting state(S) for both algorithms is up or right action

Sum of rewards per episode (averaged over number of experiments)

Results:1000 episodes

Parameters:

$\gamma = 0.95$

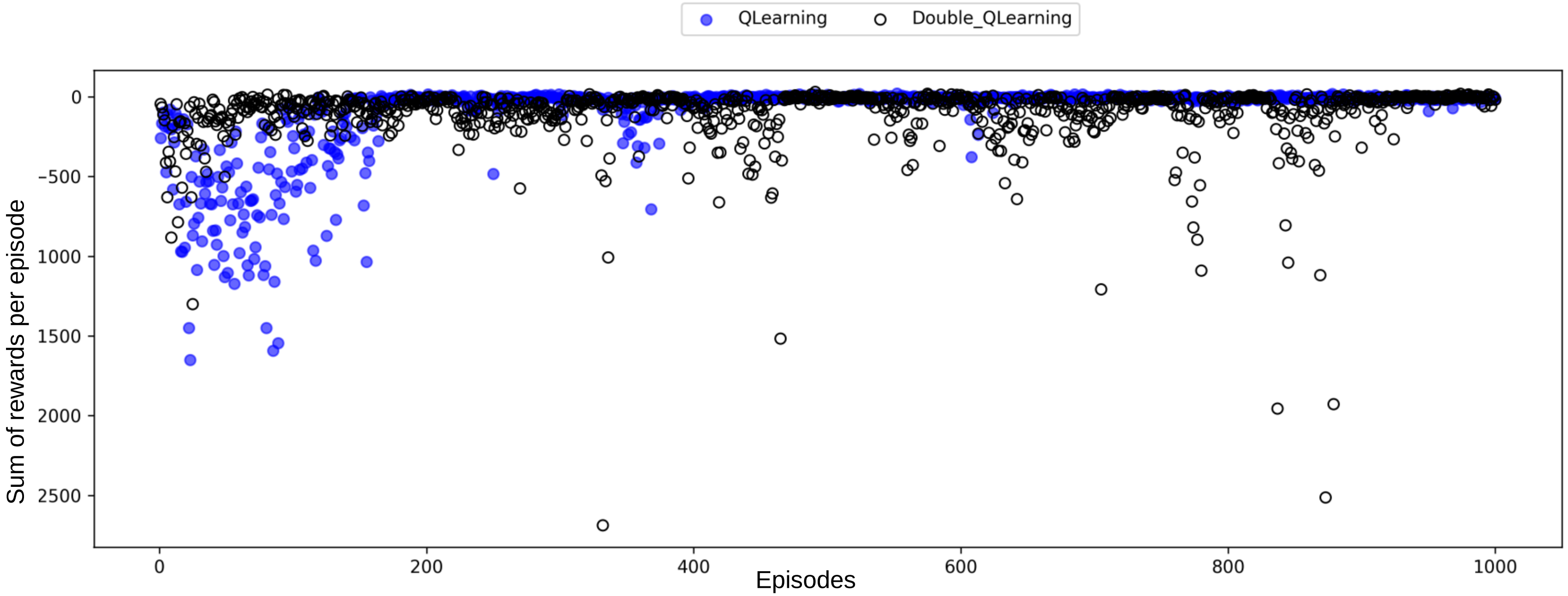
$number\ of\ trials = 100$

$\varepsilon = \frac{1}{\sqrt{n(s)}}$

$n(s) = number\ of\ times\ state\ s\ has\ been\ visited$

$\alpha = \frac{1}{n(s,a)^{0.8}}$

$n(s,a) = number\ of\ updates\ for\ each\ state, action$



Result in grid world for Q-learning and Double Q-learning: Sum of rewards per episode (averaged over number of experiments)

Double Q-learning as measured in (average) sum of rewards per episode performs better than that of Q-learning in the first few episodes in this setting.

Sum of rewards per episode (averaged over number of experiments)

Results:1000 episodes

Parameters:

$\gamma = 0.95$

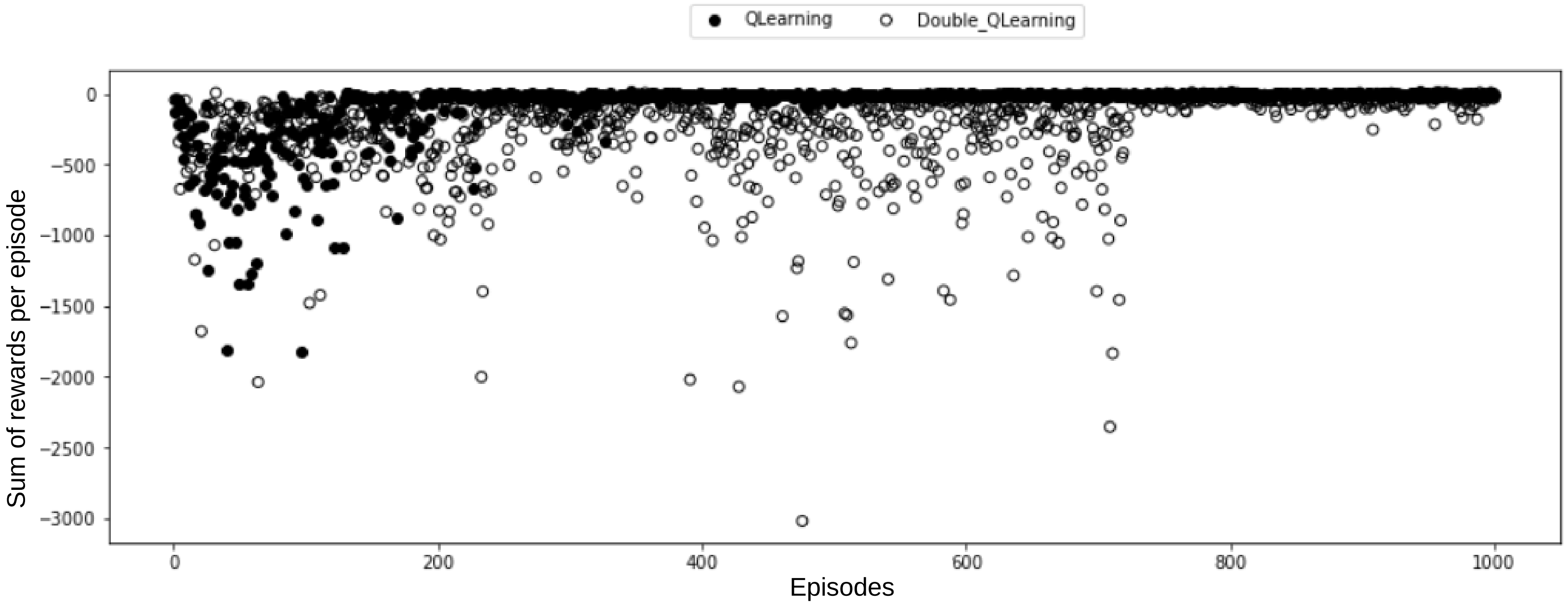
$number\ of\ trials = 100$

$$\varepsilon = \frac{1}{\sqrt{n(s)}}$$

$n(s) = number\ of\ times\ state\ s\ has\ been\ visited$

$$\alpha = \frac{1}{n(s,a)^{0.8}}$$

$n(s,a) = number\ of\ updates\ for\ each\ state, action$

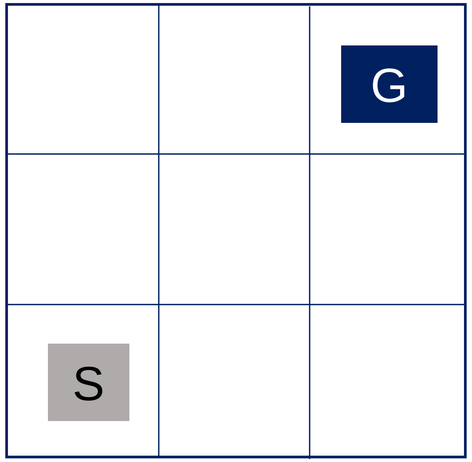


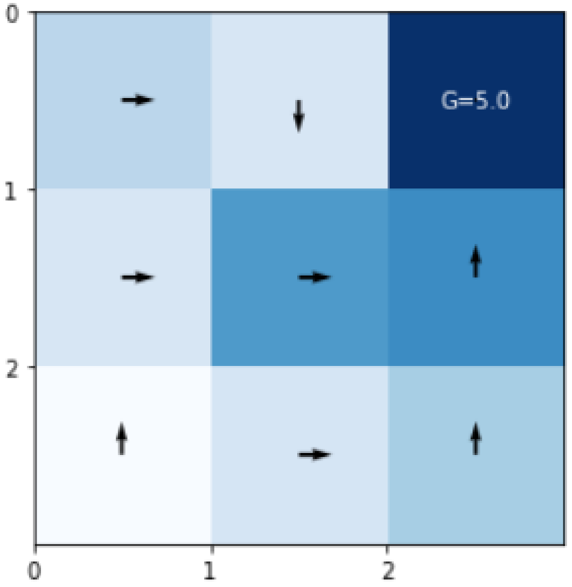
Result in grid world for Q-learning and Double Q-learning: Sum of rewards per episode (averaged over number of experiments)

Double Q-learning as measured in (average) sum of rewards per episode performs better than that of Q-learning in the first few episodes in this setting.

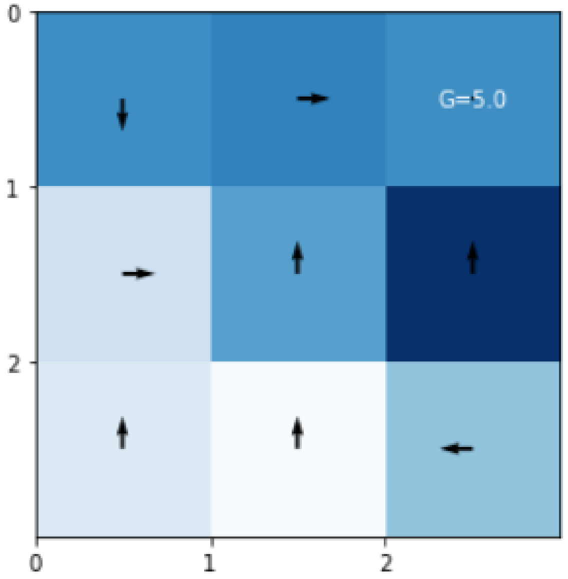
Snapshot of a Policy

Results: 1000th episode





Q-learning



Double Q-learning

Action from the starting state(S) for both algorithms is up or right action

Grid World

Parameters

Explore more and take larger steps (larger ε and α) at the beginning, but as time goes by, we want to explore less and take smaller steps because we will be focusing more close to the goal

1. Constant ε for ε-greedy policies is not good
2. Constant learning rate α is not good

$$\varepsilon = \frac{1}{\sqrt{n(s)}}$$

$$n(s) = number\ of\ times\ state\ s\ has\ been\ visited$$

$$\alpha = \frac{1}{n(s,a)^{0.8}}$$

$$n(s,a) = number\ of\ updates\ for\ each\ state, action$$

Conclusion

Q-learning can sometimes perform poorly for problems in which multiple actions yield stochastic, overlapping rewards.

Double Q-learning can be used as an alternative for Q-learning, since it can perform quite well in settings in which Q-learning suffers from overestimations.

References

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Hasselt, H. (2010). Double Q-learning. *Advances in neural information processing systems*, *23*.