# Reinforcement Learning Assignment 2

Student Number:13064947 - John Goodacre

February 15, 2018

## 1 Introduction

This document gives chart outputs and answers questions for the assignment. For associated code implementations please see the Jupyter notebook. I have attached chart outputs at the end.

## 2 Policy Improvement

### 2.1 One time Greedy policy improvement- Is this optimal?

If we imagine three policies, $\pi^{rand}$ our initial random policy, $\pi^{greed}$ our new policy after performing policy evaluation followed by one greedy policy improvement and finally $\pi^*$ the optimal policy. Then it is very clear that in general one greedy policy improvement will not lead to an optimal policy. To be clear when we perform policy evaluation, we are performing it with respect to a specific policy - in this case the random policy. By subsequently altering our policy in a greedy fashion we do guarantee a policy improvement over the prior policy, but there are no guarantees whatsoever that this will happen to coincide with the optimal policy.

That is the general case. However in simple specific cases, such as the maze case in this example it could well be that all that is needed is one policy evaluation (to some form of stability of state values), followed by one greedy improvement and that they happen to coincide. But as I said this is not true in general.

I tested this empirically. For the number of iterations given in the notebook, this is not the case and does not lead to an optimal policy. I subsequently reran for a few billion iterations and again this was not the case. I do have a sneaky suspicion that for this simple maze example, if we ran for a long enough time that this could actually be case, that even for a random starting policy with one greedy policy improvement it may immediately take us to an optimal policy

(indeed a smaller example in the lectures did precisely this). But to point out once again, this is most definitely not the case in general.

### 2.2 Full policy evaluation and improvement - will be policy become optimal?

Yes it will. If one thinks through the steps, each time we perform a policy evaluation and policy improvement we end up with a new policy $\pi^{new}$, we know that for all states s, $v_{\pi^{new}(s)} \geq v_{\pi(s)}$. Policy evaluation always converges, so each policy improvement we either have that $\pi^{new} > \pi$, or $\pi^{new} = \pi$. The former is a policy improvement, when we have reached the latter (equality), we have simply restated the Bellman optimality equation and thus have achieved an optimal policy.

## 3 Q-learning Implementation - Analyse results

### 3.1 Q-Learning v Sarsa - qualitative differences

If one considers the greedy policy after training, the first observation is that Sarsa takes a different path to the goal. It travels down under the barrier and then back up. Q-Learning takes a shorter path going up over the barrier and back down the the goal. Also the way that the agents went around the barrier differed. Sarsa took a wider path, whereas Q-Learning was tight to the barrier. I will explain why in the next section.

### 3.2 Why do the policies differ in this way?

First note that Sarsa is an on-policy algorithm. When the q(s,a) values are updated, they are updated based upon the actual behavioural policy being executed. In this case an $\epsilon$-Greedy behaviuoral policy. This explains why Sarsa takes both a different path and a wider path compared to Q-Learning.

Recall that if either algorithm was next to a wall, then the $\epsilon$-Greedy policy will explore and sometimes that exploration will result in a penalty (due to the penalty of hitting the wall). Sarsa is evaluating this exact policy and so reduces the chance of this possibility occurring by taking a safer route (note this is in this example, clearly if discounting was more punitive and the end goal reward much higher relative to the penalty of hitting the wall then things could be different).

In contrast Q-learning is an off-policy algorithm. It may well be executing the same behvioural policy (in this case) as Sarsa i.e. $\epsilon$-Greedy. However, it is learning the values of a different target policy. Greedy. Thus, Q-Learning is learning the values of the Greedy policy, despite executing a different policy. In this case it quickly learns the optimal route to the goal and maximises q-values by taking the shortest path to negate discounting.

### 3.3 Which greedy policy is better, in terms of actual value?

The Q-learning algorithm is better, it takes a shorter path to the goal, and if one examines the q-values they themselves are higher than for Sarsa. This is of course because it is evaluating the greedy policy and then executes it at the end. Sarsa is both executing and evaluating an $\epsilon$-Greedy policy, and at the end operates a greedy policy.

Note that things could be different if we introduced uncertainty into the equation where there is a non-zero probability that the action taken does not lead to the next state that one expects (i.e. one could be blown into a wall), then in this case the Sarsa algorithm could perform better.

## 4 Noisy environments - Why does Double-Q have higher reward

### 4.1 The dynamics of Double-Q learning

Q-learning has a positive bias that is introduced because it uses the maximum action value as an approximation for the maximum expected action value and is thus more likely to select over-estimated values. This can be particularly problematic in noisy environments. Double-Q estimates two different policies, greedy with respect to the other. It selects according to one, and evaluates according to the other and randomly chooses which to update. However, when acting it can use both.

### 4.2 How this affects behaviour

The key difference in behaviour is as regards removing the upward bias in the estimation of action values (although I note that the original paper states that Double-Q can still under-estimate action values and is not the end of the story).

### 4.3 Why does this yield higher rewards

I examined the action values for both policies after training. By the time we stopped training, greedy policy with respect to Double-Q would indeed have been optimal. However the Q action values were far more noisy and a greedy policy by this stage is not yet optimal and it would have required more training. (Clearly this is for the reasons given above).

Thus, the greedy policy for Double-Q is more accurate in this example, at the same stage after a number iterations and thus the actions taken under the $\epsilon$-Greedy behaviour in training will also on average result in higher rewards.

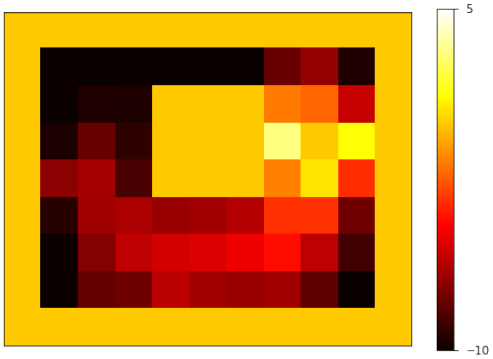
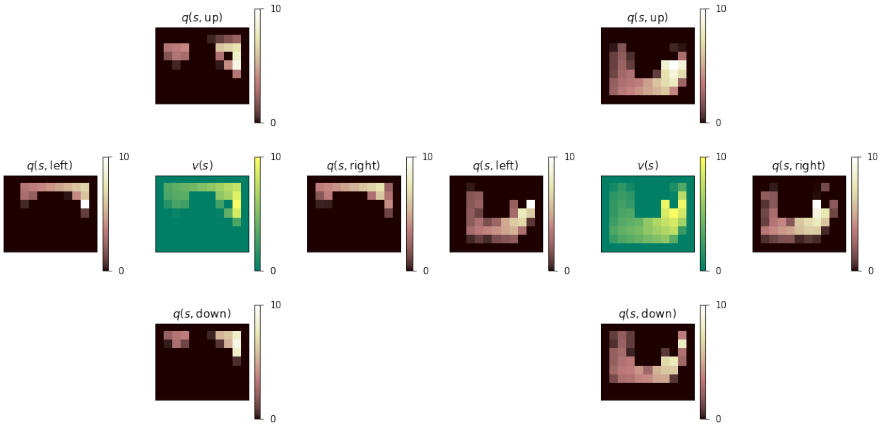
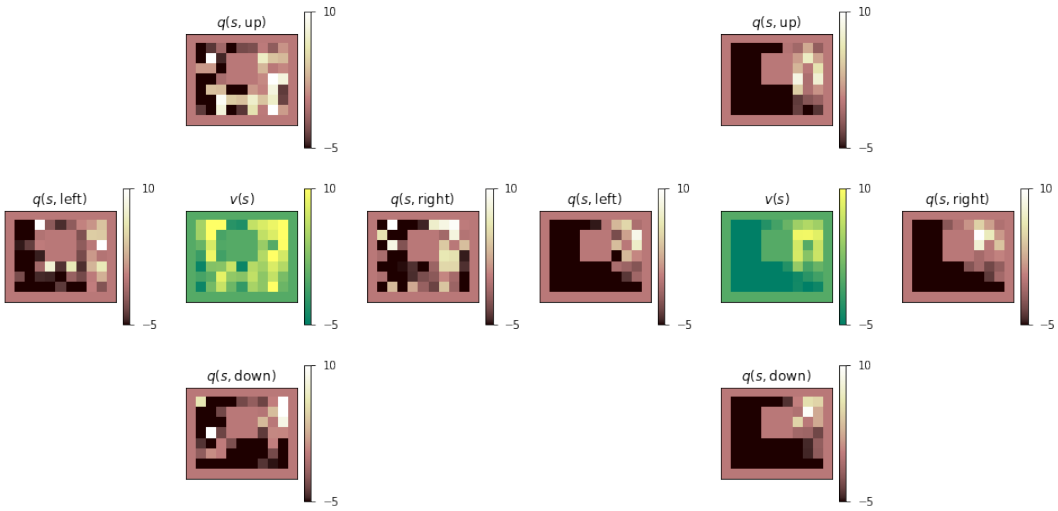



Figure 1: TD learning - random agent state values




(a) Q learning (b) Sarsa

Figure 2: Q learning and Sarsa, action values




(a) Q learning Agent (b) Double-Q learning agent

Figure 3: Comparison of Q and Double-Q action values

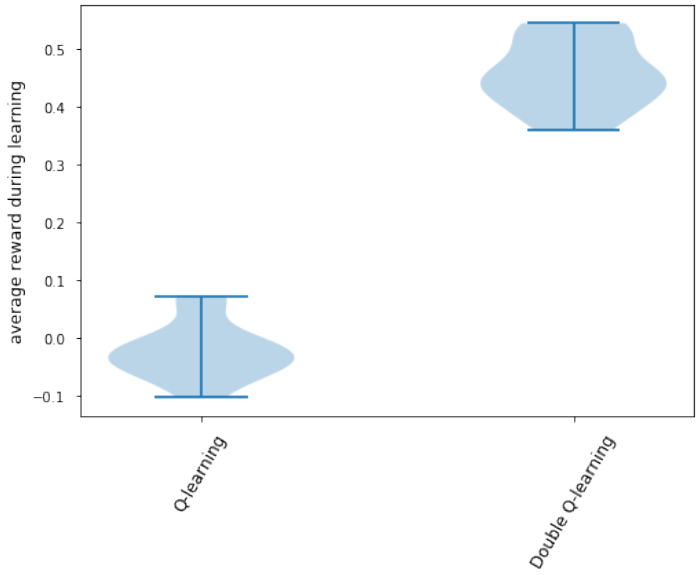



Figure 4: Comparison of rewards Double-q versus q