Temporal difference learning has been used to estimate the value function. We now want to obtain an estimation of the optimal policy itself. To do this, a TD learning style update is used which is non stationary and non-episodic. However, there are 2 types of policy approaches that can be used for this, SARSA and Q-learning and they are explained below:

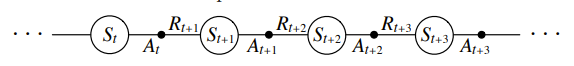
Fully written out as state, action, reward, state, action, SARSA is an on policy algorithm which means the policy it uses to select an action A’ is the same policy used to update the Q-function . The name SARSA comes from the sequence taken every episode when learning the action value function as shown in figure 1.0.

Figure 1.0 – episode of alternating sequence of states and state-action pairs

The algorithm for SARSA itself to update the state action-pair is shown in equation (1). Where is the expected return for taking an action in state at time . is the constant weighting that denotes learning rate and is the correction signal. The update for the Q-function is done for every part from the terminal state in which is equal zero instead.

(1)

Figure 1.1 shows the pseudo code for SARSA algorithm. The state action values are initially defined for all states along with the algorithm parameters for the learning rate and which is used to define the probability of exploration depending on the policy one wishes to use such as -greedy or -soft policies.

A loop is then generated for every episode, that starts by taking an action depending on the chosen policy (usually -greedy). A nested loop for each step in this episode is then created where because SARSA is on policy, the policy , is used to take the action . The state and reward is then recorded. Using the chosen policy again, an action is selected before all obtained values are finally input into equation (1). This repeats until the final state is terminal.

The result of this is SARSA will converge with a probability of 1 to an absolute optimal policy and action-value function if all state-action pairs are visited an infinite number of times. In addition to this, since the system uses a greedy policy, it will also converge if exploration is decreased over time for example, .

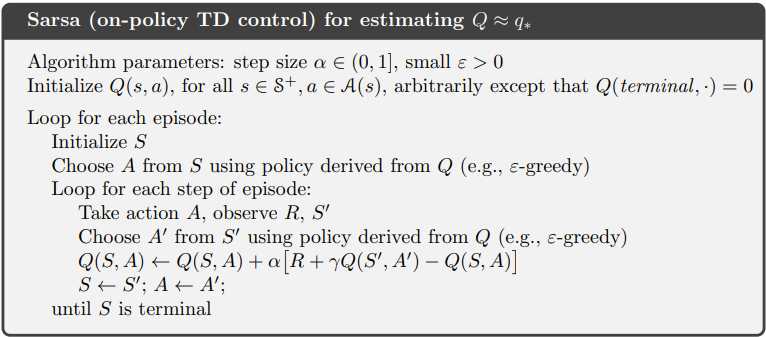


Figure 1.1 – SARSA pseudo code

Q-learning on the other hand is an off policy algorithm. The algorithm for it is shown in equation (2). The use of the function is what makes the algorithm off policy because instead of updating the action function Q using the action taken ’, it uses the greedy action , i.e. the value from the best possible action.

(2)

Figure 1.2 shows the pseudo code for Q-learning. The initialising of parameters, the looping of episodes and steps within each episode and choosing/taking actions based off greedy policy is all similar to SARSA. However with the function, it changes the way the Q-function is updated so this makes it seem as if the greedy policy is followed completely or and there is no exploration.

As a result, given an infinite amount of time and episodes to experience and learn, Q-learning will directly converge with probability of 1 to an optimal policy.

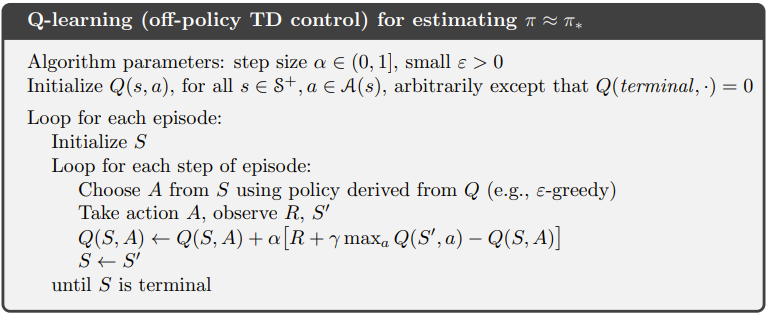


Figure 1.2– Q-learning pseudo code