11.3.6 On-Policy Learning

Q-learning learns an optimal policy no matter what the agent does, as long as it explores enough. There may be cases where ignoring what the agent actually does is dangerous (there will be large negative rewards). An alternative is to learn the value of the policy the agent is actually carrying out so that it can be iteratively improved. As a result, the learner can take into account the costs associated with exploration.

An off-policy learner learns the value of the optimal policy independently of the agent's actions. Q-learning is an off-policy learner. An on-policy learner learns the value of the policy being carried out by the agent, including the exploration steps.

```
controller SARSA(S,A,γ,α)
inputs:
        S is a set of states
        A is a set of actions
        γ the discount
        α is the step size
internal state:
        real array Q[S,A]
        previous state s
        previous action a
begin
        initialize Q[S,A] arbitrarily
        observe current state s
        select action a using a policy based on Q
        repeat forever:
                carry out an action a
                observe reward r and state s'
                select action a' using a policy based on Q
                Q[s,a] ←Q[s,a] + α(r+ γQ[s',a'] - Q[s,a])
                s ←s'
                a ←a'
        end-repeat
end
```

Figure 11.13: SARSA: on-policy reinforcement learning

SARSA (so called because it uses state-action-reward-state-action experiences to update the Q-values) is an on-policy reinforcement learning algorithm that estimates the value of the policy being followed. An experience in SARSA is of the form □s,a,r,s',a'□, which means that the agent was in state s, did action a, received reward r, and ended up in state s', from which it decided to do action a'. This provides a new experience to update Q(s,a). The new value that this experience provides is $r+\gamma Q(s',a')$.

Figure 11.13 gives the SARSA algorithm.

SARSA takes into account the current exploration policy which, for example, may be greedy with random steps. It can find a different policy than Q-learning in situations when exploring may incur large penalties. For example, when a robot goes near the top of stairs, even if this is an optimal policy, it may be dangerous for exploration steps. SARSA will discover this and adopt a policy that keeps the robot away from the stairs. It will find a policy that is optimal, taking into account the exploration inherent in the policy.

Example 11.12: In Example 11.10, the optimal policy is to go up from state s_0 in Figure 11.8. However, if the agent is exploring, this may not be a good thing to do because exploring from state s_2 is very dangerous.

If the agent is carrying out the policy that includes exploration, "when in state s, 80% of the time select the action a that maximizes Q[s,a], and 20% of the time select an action at random," going up from s_0 is not optimal. An on-policy learner will try to optimize the policy the agent is following, not the optimal policy that does not include exploration.

If you were to repeat the experiment of Figure 11.11, SARSA would back up the -1 values, whereas Q-learning did not because actions with an estimated value of 0 were available. The Q-values in parts (a) and (b) of that figure would converge to the same values, because they both converge to the value of that policy.

The Q-values of the optimal policy are less in SARSA than in Q-learning. The values for SARSA corresponding to part (c) of Figure 11.11, are as follows:

Iteration	$Q[s_0,\text{right}]$	$Q[s_1,\text{upC}]$	$Q[s_3,\text{upC}]$	$Q[s_5,\text{left}]$	$Q[s_4,\text{left}]$
∞	9.2	10.1	12.7	15.7	18.0

The optimal policy using SARSA is to go right at state s_0. This is the optimal policy for an agent that does 20% exploration, because exploration is dangerous. If the rate of exploration were reduced, the optimal policy found would change. However, with less exploration, it would take longer to find an optimal policy.

SARSA is useful when you want to optimize the value of an agent that is exploring. If you want to do offline learning, and then use that policy in an agent that does not explore, Q-learning may be more appropriate.