Reinforcement Learning: Homework 1

Jorrim Prins
Student nr.: 11038934
jorrimprins.prins@student.uva.nl

Yke Rusticus
Student nr.: 11306386
yke.rusticus@student.uva.nl

2.1 Dynamic Programming

1. The value function is given by:

$$v^\pi(s) = \mathbb{E}[G_t|S_t = s] \tag{1}$$
$$= \mathbb{E}_{a\sim\pi}[\mathbb{E}_\pi[G_t|S_t = s, A_t = a]] \tag{2}$$
$$= \mathbb{E}_{a\sim\pi} q^\pi(s, a). \tag{3}$$

In the stochastic case, the expectation value is given by

$$v^\pi(s) = \sum_a \pi(a|s) q^\pi(s, a). \tag{4}$$

In the deterministic case, where $\pi(s) = a$, we have

$$v^\pi(s) = q^\pi(s, a). \tag{5}$$

2. Eq. 4.10 in the book:

$$v_{k+1}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, A_t = a] \tag{6}$$
$$= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')]. \tag{7}$$

For the Q-value iteration update we provide both state s and action a to the Q-function, so we do not need to take the maximum value w.r.t a anymore. In other words, we lose "$\max_a$". However, since we need to incorporate the Q-value of the next state-action, we get a new variable a', i.e. the action we will take in the next state. Since we want to maximize the state-value function we thus need to maximize the next state-action value w.r.t a'. The Q-value iteration update will therefore be:

$$q_{k+1}(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_k(s', a')]. \tag{8}$$

3.

Algorithm 1: Policy Evaluation in terms of $Q^\pi(s, a)$ instead of $V^\pi(s)$

Loop:
  $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
      Loop for each $a \in \mathcal{A}(s)$: ($\mathcal{A}(s)$: set of possible actions in state s)
        $q \leftarrow Q(s, a)$
        $Q(s, a) \leftarrow \sum_{s',r} p(s', r|s, a)[r + \gamma Q(s', \pi(s'))]$
        $\Delta \leftarrow \max(\Delta, |q - Q(s, a)|)$
until $\Delta < \theta$ (a small positive number determining the accuracy of estimation).

4.

Algorithm 2: Policy Improvement in terms of $Q^{\pi}(s, a)$ instead of $V^{\pi}(s)$

policy-stable $\leftarrow true$
For each $s \in \mathcal{S}$:
    old-action $\leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_a Q(s, a)$
    If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
If *policy-stable*, then stop and return $Q \approx q_*$ and $\pi \approx \pi_*$; else go to Policy Evaluation

2.2 Coding Assignment - Dynamic Programming

2. Policy iteration only iterates over the available states in an environment, to find the most 'profitable' action in each state and update the policy. Q-value iteration, on the other hand, integrates the consequences of actions from the next state and therefore one could expect it needs more backup operations than the policy iteration algorithm. However, since policy iteration switches back and forth between policy evaluation and improvement, we found that it needed more update steps to converge than our implemented Q-value iteration algorithm. The reason for the efficiency of the Q-value algorithm is that it learns the policy implicitly while updating the Q-value function independently of the policy. So in this case there is no need to switch back and forth between evaluation and improvement.