7.6 Q(λ)

Two different methods have been proposed that combine eligibility traces and Q-learning; we call them *Watkins's Q*(λ) and *Peng's Q*(λ), after the researchers who first proposed them. First we describe Watkins's Q(λ).

Recall that Q-learning is an off-policy method, meaning that the policy learned about need not be the same as the one used to select actions. In particular, Q-learning learns about the greedy policy while it typically follows a policy involving exploratory actions--occasional selections of actions that are suboptimal according to Q_t. Because of this, special care is required when introducing eligibility traces.

Suppose we are backing up the state-action pair s_t, a_t at time t. Suppose that on the next two time steps the agent selects the greedy action, but on the third, at time $t+3$, the agent selects an exploratory, nongreedy action. In learning about the value of the greedy policy at s_t, a_t we can use subsequent experience only as long as the greedy policy is being followed. Thus, we can use the one-step and two-step returns, but not, in this case, the three-step return. The n-step returns for all $n \geq 3$ no longer have any necessary relationship to the greedy policy.

Thus, unlike TD(λ) or Sarsa(λ), Watkins's Q(λ) does not look ahead all the way to the end of the episode in its backup. It only looks ahead as far as the next exploratory action. Aside from this difference, however, Watkins's Q(λ) is much like TD(λ) and Sarsa(λ). Their lookahead stops at episode's end, whereas Q(λ)'s lookahead stops at the first exploratory action, or at episode's end if there are no exploratory actions before that. Actually, to be more precise, one-step Q-learning and Watkins's Q(λ) both look one action *past* the first exploration, using their knowledge of the action values. For example, suppose the first action, a_{t+1}, is exploratory. Watkins's Q(λ) would still do the one-step update of $Q_t(s_t, a_t)$ toward $r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a)$. In general, if a_{t+n} is the first exploratory action, then the longest backup is toward

$$r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n \max_a Q_t(s_{t+n}, a),$$

where we assume off-line updating. The backup diagram in Figure 7.13 illustrates the forward view of Watkins's Q(λ), showing all the component backups.

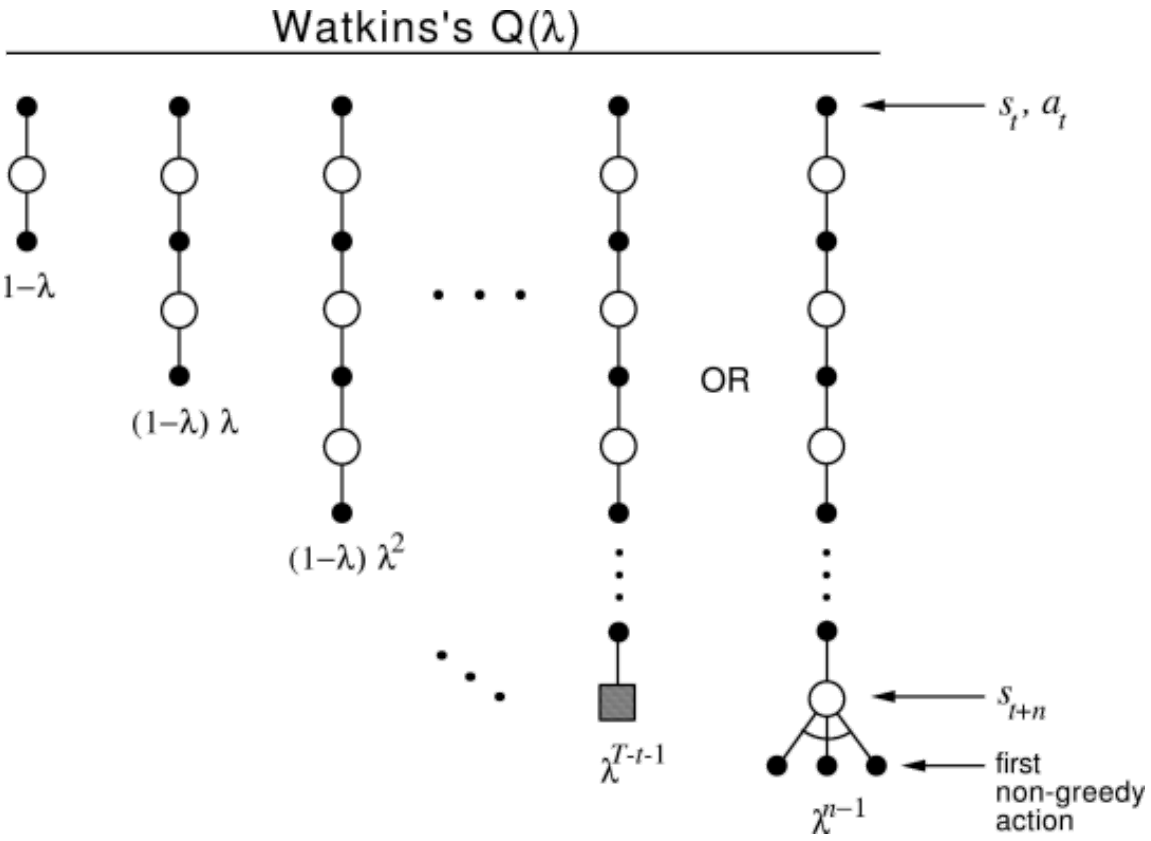

Figure 7.13: The backup diagram for Watkins's Q(λ). The series of component backups ends either with the end of the episode or with the first nongreedy action, whichever comes first.

The mechanistic or backward view of Watkins's Q(λ) is also very simple. Eligibility traces are used just as in Sarsa(λ), except that they are set to zero whenever an exploratory (nongreedy) action is taken. The trace update is best thought of as occurring in two steps. First, the traces for all state-action pairs are either decayed by $\gamma\lambda$ or, if an exploratory action was taken, set to 0. Second, the trace corresponding to the current state and action is incremented by 1. The overall result is

$$e_t(s,a) = \mathcal{I}_{ss_t} \cdot \mathcal{I}_{aa_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{if } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a); \\ 0 & \text{otherwise,} \end{cases}$$

where, as before, $\mathcal{I}_{xy}$ is an identity indicator function, equal to 1 if $x = y$ and 0 otherwise. The rest of the algorithm is defined by

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a),$$

where

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t).$$

Figure 7.14 shows the complete algorithm in pseudocode.

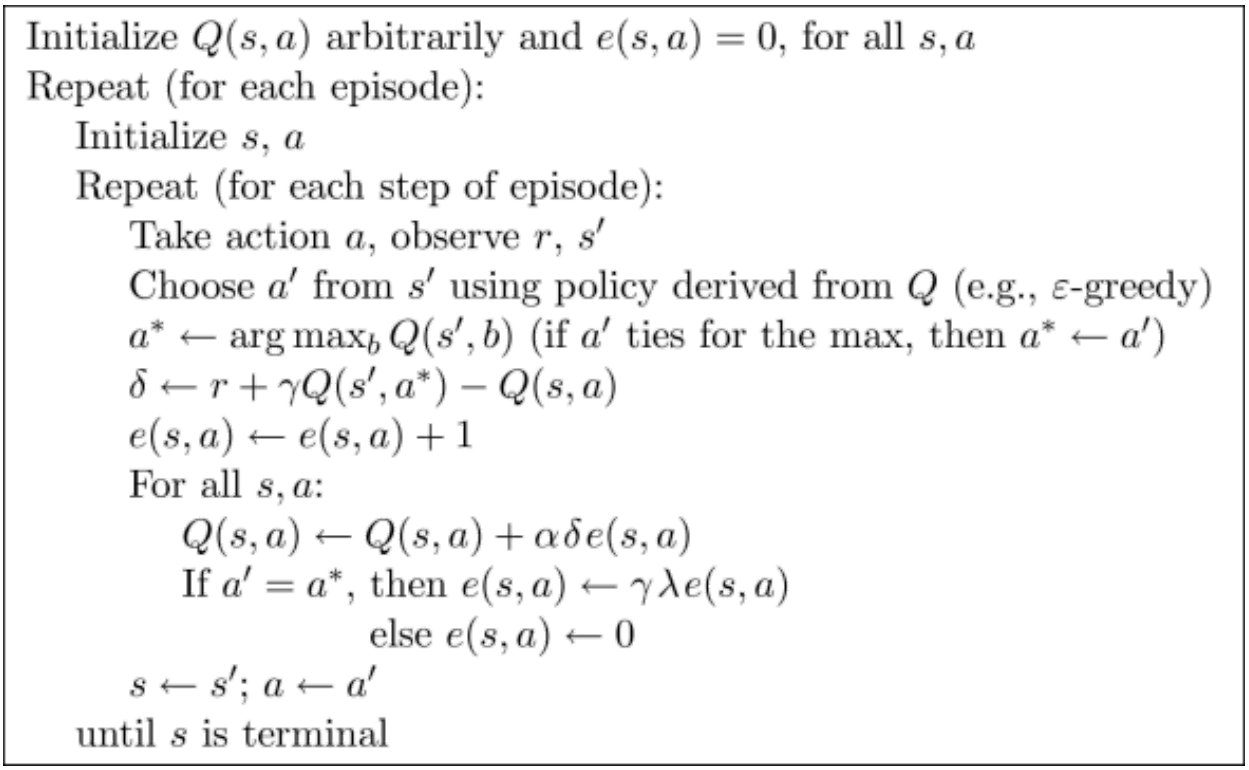

Figure 7.14: Tabular version of Watkins's Q(λ) algorithm.

Unfortunately, cutting off traces every time an exploratory action is taken loses much of the advantage of using eligibility traces. If exploratory actions are frequent, as they often are early in learning, then only rarely will backups of more than one or two steps be done, and learning may be little faster than one-step Q-learning. Peng's Q(λ) is an alternate version of Q(λ) meant to remedy this. Peng's Q(λ) can be thought of as a hybrid of Sarsa(λ) and Watkins's Q(λ).

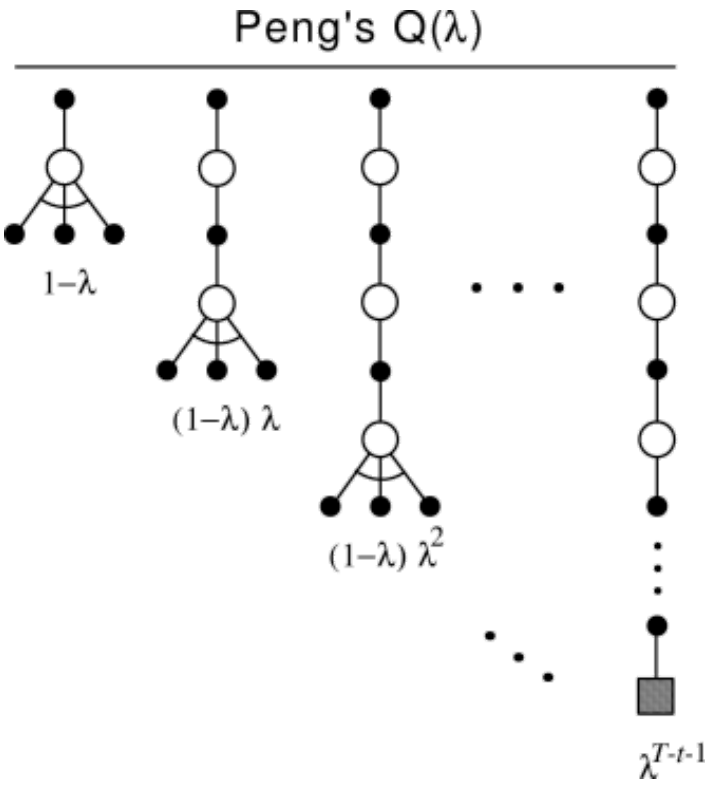

Figure 7.15: The backup diagram for Peng's Q(λ).

Conceptually, Peng's Q(λ) uses the mixture of backups shown in Figure 7.15. Unlike Q-learning, there is no distinction between exploratory and greedy actions. Each component backup is over many steps of actual experiences, and all but the last are capped by a final maximization over actions. The component backups, then, are neither on-policy nor off-policy. The earlier transitions of each are on-policy, whereas the last (fictitious) transition uses the greedy policy. As a consequence, for a fixed nongreedy policy, Q_t converges to neither Q^π nor Q^* under Peng's Q(λ), but to some hybrid of the two. However, if the policy is gradually made more greedy, then the method may still converge to Q^*. As of this writing this has not yet been proved. Nevertheless, the method performs well empirically. Most studies have shown it performing significantly better than Watkins's Q(λ) and almost as well as Sarsa(λ).

On the other hand, Peng's Q(λ) cannot be implemented as simply as Watkins's Q(λ). For a complete description of the needed implementation, see Peng and Williams (1994, 1996). One could imagine yet a third version of Q(λ), let us call it *naive Q*(λ), that is just like Watkins's Q(λ) except that the traces are not set to zero on exploratory actions. This method might have some of the advantages of Peng's Q(λ), but without the complex implementation. We know of no experience with this method, but perhaps it is not as naive as one might at first suppose.