Principles of Artificial Intelligence
Reinforcement Learning

Vasant Honavar
Artificial Intelligence Research Laboratory
Pennsylvania State University

©Vasant Honavar

1 Reinforcement Learning

1.1 Background Information

Consider an agent possessing a set of sensors through which it can observe the state of the environment and a set of effectors by means of which it can perform actions to alter the state of the environment. In this section we shall explore the question of how such an agent can learn *successful policies* by experimenting with the environment, and more specifically, how the provision of *rewards* can be used to facilitate learning (the italicized terms will be made more precise later).
Let us begin by noting some of the factors which can make the learner's task difficult in such situations:

- **The credit assignment problem.** This refers to the fact that the optimal choice of action is generally unknown a priori and must be inferred on the basis of rewards which may be delayed. It is often difficult to determine which of a long sequence of actions were primarily responsible for its ultimate success or failure (for example, which particular moves in a game were most critical to eventually winning or losing).

- **The exploration vs. exploitation dilemma.** In the real world, there is often a tradeoff between exploiting current knowledge to achieve predictable results and exploring the environment in the hope of gaining additional knowledge which can improve results in the future. It may be difficult for an agent to determine the appropriate balance between these two general approaches to action.

- **Inherent limitations to knowledge.** In some cases the environment may be only partially observable or may be highly dynamic, independent of the agent's actions. It may also be stochastic, either relative to the agent's incomplete knowledge of it or in some innately nondeterministic sense. The agent's actions may also be less than completely reliable (e.g., a robot's gripper may not always succeed in grasping the desired object). All of these can lead in obvious ways to difficulties in determining what course of action to take under a particular set of circumstances.

In the material which follows, we will make the simplifying assumption that the agent can reliably determine the current state of the environment, but should bear in mind that this is by no means always the case.

1.2 Definitions

Given a set S of possible states and a set A of possible actions, we define a **policy** to be a mapping $\Pi : S \times A \rightarrow A$ which specifies which action should be taken out of all possible actions when the environment is in a specified state. Although the set of actions which could possibly be taken can vary with the state, we shall assume for simplicity that it does not, so that we may think of a policy as a simple mapping $\Pi : S \rightarrow A$ of states to actions. Since each action specified by a policy may result in either a reward or a transition to a new state or both, what we are seeking is a policy which in some sense optimizes these outcomes of the specified actions.

To formalize this notion of optimizing a policy, we define the **cumulative discounted reward** for a policy Π as follows:

$$V^{\Pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

where s_t and r_t are the state of the environment and the reward received at time t, respectively, and γ is a **discounting factor** greater than 0 and less than 1. Intuitively, the cumulative discounted reward is a measure of the relative value of the choice of actions specified by the policy starting at time t, with the value of future rewards discounted exponentially by a factor γ. Note that this definition assumes that the rewards r_t are bounded by a constant so that the infinite series converges.
Our goal can now be formalized as finding a policy Π^* which maximizes the cumulative discounted reward, i.e. one for which

$$\forall s \in S, \Pi^* = \arg\max_{\Pi} V^{\Pi}(s)$$

where $\arg\max_x f(x)$ denotes the value of x for which $f(x)$ is maximal.

1.3 Learning Evaluation Functions

If at each time step the learner were to receive immediate, reliable rewards for the action just taken, the task of learning an optimal Π^* would be relatively simple, and could be accomplished by using one of the gradient climbing techniques we have already studied. The task becomes more difficult when the only training information available to the learner is the sequence of states resulting from its actions and the corresponding rewards, scattered over time.
One approach to overcoming this difficulty is to try learning an **evaluation function** (analogous to a heuristic) which evaluates each action available in a given state and returns a numerical score indicating the relative utility of performing that action in that state. We start by defining

$$V^*(s) = V^{\Pi^*(s)}$$

for any state $s \in S$, and denoting by $r(s, a)$ the reward obtained after taking action a in state s and by δ the state transition function (i.e., the mapping $\delta : S \times A \to A$ which maps a state s and an action a to the next state which results from taking action a while in state s). We can then give an equivalent definition for the optimal policy as follows:

$$\Pi^* = \arg\max_a [r(s, a) + \gamma V^*(\delta(s, a))]$$

In this case we express the optimal policy Π^* as that which maximizes the sum of the immediate reward and the cumulative discounted reward from the resulting state. We can them attempt to learn the evaluation function V^* by some direct means and then deriving Π^* from V^* with this equation.
Although this approach to the problem has dominated research in reinforcement learning until recently, it proves difficult to implement effectively in practice since neither the reward function r nor the state transition function δ is generally known a priori. In the next section we shall see how a simple restatement of the problem can help to overcome this obstacle.

1.4 Q-Learning

In a 1989 Ph.D. thesis which has subsequently formed the basis for much current research, C.J. Watkins observed that if we 'hide' the details of the reward and state transition functions in an aggregate evaluation function $Q : S \times A \to A$ defined by $Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$, then we can learn Q directly by a relatively simple iterative approximation algorithm and then calculate Π^* from Q by rewriting our definition of an optimal policy as

$$\Pi^* = \arg\max_a Q(s, a)$$

Watkins' key insights were that (1) $Q(s, a)$ summarizes in one number all the relevant information about the immediate as well as the future consequences of performing an action in a given state, and (2) the iterative approximation approach to learning Q directly is provably convergent in the limit (provided we assume a deterministic model in which both $r(s, a)$ and $\delta(s, a)$ are uniquely determined by s and a).

To construct the **Q-learning algorithm**, we first observe that

$$V^*(s) = \max_a Q(s,a) \forall s \in S$$

by the definition of Q. Thus,

$$\begin{aligned} Q(s,a) &= r(s,a) + \gamma V^*(\delta(s,a)) \\ &= r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a') \end{aligned}$$

$\forall s \in S, a \in A$. We can then base the algorithm upon this recurrence relation:

```
1. $\hat{Q}(s,a) \leftarrow 0 \, \forall s \in S, a \in A$
2. do forever
    2a. observe the current state $s$
    2b. select and execute an action $a$
    2c. receive a reward $r(s,a)$
    2d. observe the resulting state $s' = \delta(s,a)$
    2e. $\hat{Q}(s,a) \leftarrow r(s,a) + \gamma \max_{a'} \hat{Q}(s',a')$
```

2 Reinforcement Learning (continued)

2.1 Examples

To further illustrate reinforcement learning, some examples are provided:

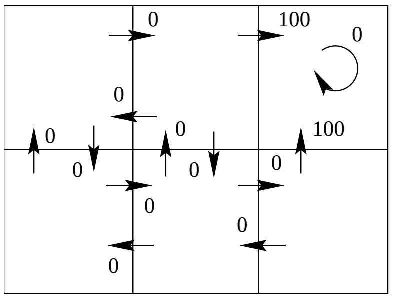

Figure 1: Initial configuration of the rewards

squares: states
arrows: actions available
numbers: $r(s,a)$ (reward)

Follows from $Q(s,a) = \gamma(s,a) + \gamma V^*(\delta(s,a))$, we get the Q value table (Figure 2), where $\gamma = 0.9$. The corresponding optimal policies for each state is shown in Figure 3.
Figure 4 gives the initial state s_1 and relevant $\hat{Q}$ values. The next state s_2 is computed by $\hat{Q}(s_1, a_{right}) = \gamma(s_1, a_{right}) + \gamma max\{\hat{Q}(s_2, a')\}$, and is shown in Figure 5, where $r = 0.9$.

2.2 Convergence of Q-learning

Theorem Consider a Q-learning in a deterministic Markovian environment with bounded rewards, that is $\forall(s,a)$, $r(s,a) < C$ (C is a finite positive constant). Suppose the learner uses the update rule: $\hat{Q}(s,a) = \gamma(s,a) + \gamma max\{\hat{Q}(s',a')\}$, where s, a, s', a', r, γ have the usual meaning. If each state-action pair (s,a) is visited infinitely often (in the limit), then $\hat{Q}_n(s,a)$, the Q estimates after n iterations approaches $Q(s,a)$, $\forall(s,a)$ as $n \to \infty$.

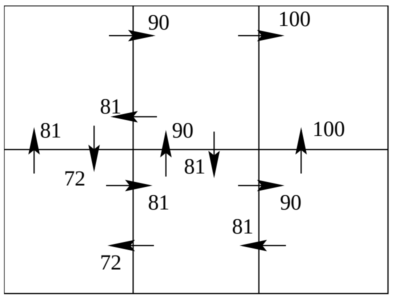

Figure 2: $Q(s, a)$ values

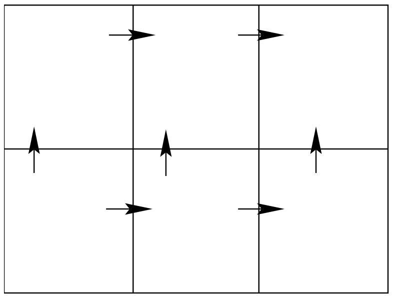

Figure 3: Optimal Policies

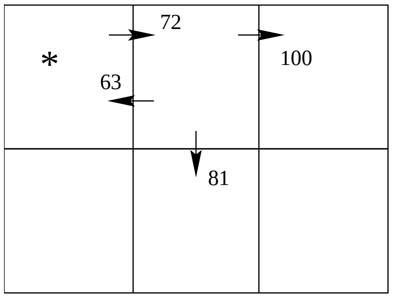

Figure 4: Initial state and corresponding $\hat{Q}$ values.

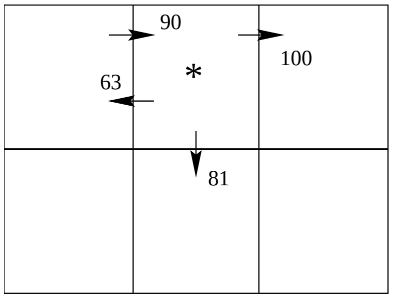

Figure 5: Next state and new $\hat{Q}$ values

Proof Since each state-action pair is tried infinitely often, consider consecutive time intervals during which each state-action pair is tried at least once. The key idea behind the proof is to show that the maximum error over all entires in the Q table is reduced by a factor of at least γ during each such interval.
Let $\hat{Q}_n$ be the table of Q values after n such intervals. Let Δ_n be the maximum error in $\hat{Q}_n$ (with respect to Q),

$$\Delta_n = \max_{(s,a)} |\hat{Q}_n(s,a) - Q(s,a)|$$

Let $s' = \delta(s,a)$ (the state resulting from performing action a in state s. Then,

$$\begin{aligned}
&|\hat{Q}_{n+1}(s,a) - Q(s,a)| \\
&= |(\gamma(s,a) + \gamma \max_{a'}\{\hat{Q}_n(s',a')\}) - (\gamma(s,a) + \gamma \max_{a'}\{Q(s',a')\})| \\
&= | \max_{a'}\{\hat{Q}_n(s',a')\} - \max_{a'}\{Q(s',a')\}| \\
&\leq \gamma \max_{a'} |\hat{Q}_n(s',a') - Q(s',a')| \\
&\leq \gamma \max_{(s'',a'')} |\hat{Q}_n(s'',a'') - Q(s'',a'')| \\
&\leq \gamma \Delta_n
\end{aligned}$$

Let $\Delta_0 =$ the max error in the initial Q estimate.
After k intervals, $\Delta_k \leq \gamma^k * \Delta_0$.
Thus, $\Delta_k \to 0$ as $k \to \infty$, since $\gamma < 1$.

2.3 Action Selection

Action Selection incorporates the trade off between exploration and exploitation.

$$\begin{aligned}
P_{a_i}(s) &= \text{Prob of selecting action } a_i \text{ when in state } s. \\
&= k^{\hat{Q}(s,a_i)} \div \sum_j k^{\hat{Q}(s,a_j)} \qquad\qquad k > 0
\end{aligned}$$

In order to apply Q-learning in practice, either the state space must have a manageable size (so that a Q-table can be stored) or there must be a way to approximate the Q-table using a function approximation (e.g. neural network).
It is relatively straightforward to extend Q-learning to handle environments with stochastic state transition and reward functions.