Notes for Reinforcement Learning

Mhttx

February 25, 2019

Contents

Chapter 1

Introduction

1.1 Reinforcement Learning

1.2 Elements of RL

Part I

Tabular Solution Methods

Chapter 2

Multi-armed Bandits

2.1 A k-armed Bandit Problem

We denote the action selected on time step t as A_t, and the corresponding reward as R_t. The value then of an arbitrary action a, denoted $q_*(a)$, is the expected reward given that a is selected:

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a] \tag{2.1}$$

We denote the estimated value of action a at time step t as $Q_t(a)$. We would like $Q_t(a)$ to be close to $q_*(a)$.

2.2 Action-value Methods

sample-average method: estimates $q_*(a)$ by averaging the rewards actually received:

$$Q_t(a) \doteq \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}} \tag{2.2}$$

As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$.

greedy action selection:

$$A_t \doteq arg \max_a Q_t(a) \tag{2.3}$$

ϵ-greedy action selection:

$$A_t \doteq \begin{cases} arg \max_a Q_t(a), & with \quad probability \quad 1-\epsilon \\ a \quad random \quad action, & with \quad probability \quad \epsilon \end{cases} \tag{2.4}$$

In the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that all the $Q_t(a)$ converge to $q_*(a)$.

2.3 The 10-armed Testbed

2.4 Incremental Implementation

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1} \tag{2.5}$$

$$
\begin{aligned}
Q_{n+1} =& \frac{1}{n}\sum_{i=1}^{n} R_i \\
=& \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
=& \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i) \\
=& \frac{1}{n}(R_n + (n-1)Q_n) \\
=& Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned} \tag{2.6}
$$

The general form of 2.6 is:

$$NewEstimate = OldEstimate + StepSize * [Target - OldEstimate] \tag{2.7}$$

Note that the step-size parameter (StepSize, denote as α or $\alpha_t(a)$) used in the incremental method described above changes from time step to time step, here $\alpha_t(a) = \frac{1}{n}$.

2.5 Tracking a Nonstationary Problem

The averaging methods discussed so far are appropriate for **stationary bandit problems**, that is, for bandit problems in which the reward probabilities do not change over time. **In effectively nonstationary cases it makes sense to give more weight to recent rewards than to long-past rewards.** One of the most popular ways of doing this is to use a **constant step-size parameter**:

$$Q_{n+1} \doteq Q_n + \alpha[R_n - Q_n] \tag{2.8}$$

This results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1:

$$
\begin{aligned}
Q_{n+1} =& Q_n + \alpha[R_n - Q_n] \\
=& \alpha R_n + (1-\alpha)Q_n \\
=& \alpha R_n + (1-\alpha)[\alpha R_{n-1} + (1-\alpha)Q_{n-1}] \\
=& \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
=& \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
& \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
=& (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned} \tag{2.9}
$$

Note that the weight, $\alpha(1-\alpha)^{n-i}$, given to the reward R_i depends on how many rewards ago, Accordingly, this is sometimes called an **exponential recency-weighted average**. eros Let $\alpha_n(a)$ denote the step-size parameter used to process the reward received after the $n_t h$ selection of action a. Conditions required to assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad and \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty \tag{2.10}$$

The first condition is required to guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the steps become small enough to assure convergence.

Note that both convergence conditions are met for the sample-average case, $\alpha_n(a) = \frac{1}{n}$, but not for the case of constant step-size parameter, $\alpha_n(a) = \alpha$. In the latter case, the second condition is not met, indicating that **the estimates never completely converge but continue to vary in response to the most recently received rewards. As we mentioned above, this is actually desirable in a nonstationary environment.**

2.6 Optimistic Initial Values

All the methods we have discussed so far are dependent to some extent on the initial action-value estimates, $Q_1(a)$. These methods are biased by their initial estimates. The downside is that the initial estimates become, in effect, a set of parameters that must be picked by the user, if only to set them all to zero. The upside is that they provide an easy way to **supply some prior knowledge** about what level of rewards can be expected.

Initial action values can also be used as a simple way to encourage exploration. We call this technique for encouraging exploration **optimistic initial values**. It is not well suited to nonstationary problems because its drive for exploration is inherently temporary. If the task changes, creating a renewed need for exploration, this method cannot help. Indeed, any method that focuses on the initial conditions in any special way is unlikely to help with the general nonstationary case. The beginning of time occurs only once, and thus we should not focus on it too much.

2.7 Upper-Confidence-Bound Action Selection

It would be better to select among the non-greedy actions according to their **potential for actually being optimal**, taking into account both **how close their estimates are to being maximal and the uncertainties in those estimates**.

$$A_t \doteq arg \max_a [Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}] \tag{2.11}$$

The idea of this **upper confidence bound (UCB)** action selection is that the square-root term is a measure of **the uncertainty or variance in the estimate of a s value.** The quantity being maxed over is thus a sort of upper bound on the possible true value of action a, with c determining the confidence level.

One difficulty is in dealing with nonstationary problems. Another difficulty is dealing with large state spaces, particularly when using function approximation.

2.8 Gradient Bandit Algorithms

$H_t(a)$ as a numerical preference of each action a.

$$\pi_t(a) \doteq softmax(H_t(a)) \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \tag{2.12}$$

The Bandit Gradient Algorithm as Stochastic Gradient Ascent: In exact gradient ascent, each preference $H_t(a)$ would be incremented proportional to the increments effect on performance:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \tag{2.13}$$

$$\mathbb{E}[R_t] = \sum_b \pi_t(b) q_*(b) \tag{2.14}$$

$$\begin{aligned} \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} [\sum_b \pi_t(b) q_*(b)] \\ &= \sum_b q_*(b) \frac{\partial \pi_t(b)}{\partial H_t(a)} \\ &= \sum_b (q_*(b) - X_t) \frac{\partial \pi_t(b)}{\partial H_t(a)} \end{aligned} \tag{2.15}$$

where X_t can be any scalar that does not depend on b, we can include it here because the gradient sums to zero over all the actions, $\sum_b \frac{\partial \pi_t(b)}{\partial H_t(a)} = 0$.

$$\begin{aligned}\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} =& \sum_b \pi_t(b)(q_*(b) - X_t)\frac{\partial \pi_t(b)}{\partial H_t(a)}/\pi_t(b) \\ =& \mathbb{E}[(q_*(A_t) - X_t)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)] \\ =& \mathbb{E}[(R_t - \bar{R}_t)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)]\end{aligned} \tag{2.16}$$

we have $\frac{\partial \pi_t(b)}{\partial H_t(a)} = \pi_t(b)(\mathbb{1}_{a=b} - \pi_t(a))$, and substituted it.

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}[(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a))] \tag{2.17}$$

Substituting a sample of the expectation above for the performance gradient in 2.13:

$$H_{t+1}(a) = H_t(a) + \alpha(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a)) \tag{2.18}$$

Chapter 3

Finit Markov Decision Process

3.1 The Agent-Environment Interface

1. **finite MDP:** the sets of states, actions and rewards($\mathcal{S}, \mathcal{A}, \mathcal{R}$) all have a finite number of elements.

2. **Markov property:** In this case, the random variables R_t and S_t have well defined discrete probability distributions dependent only on the preceding state and action.

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{3.1}$$

3. **state-transitation probabilities:**

$$p(s'|s, a) \doteq Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{3.2}$$

4. **expected rewards for stateaction pairs:**

$$r(s, a) \doteq \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) \tag{3.3}$$

5. **expected rewards for state-action-next-state triplets:**

$$r(s, a, s') \doteq \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r, s, a)}{p(s'|s, a)} \tag{3.4}$$

3.2 Goals and Rewards

Reward hypothesis:That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the **cumulative sum of a received scalar signal** (called reward).

3.3 Returns and Episodes

1. **episodic(episode) tasks:** when the agentenvironment interaction breaks naturally into subsequences, which we call episodes. Tasks with episodes of this kind are called episodic tasks.

2. **continuing tasks:** the agentenvironment interaction does not break naturally into identifiable episodes, but goes on continually without limit.

3. **return(simple):**

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{3.5}$$

4. **discounted return:**

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.6}$$

   where γ is a parameter, $0 \leq \gamma \leq 1$, called the **discount rate**. As γ approaches 1, the return objective takes future rewards into account more strongly; the agent becomes more farsighted.

5. **Returns at successive time steps:**

$$\begin{aligned} G_t \doteq& R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ =& R_{t+1} + \gamma (R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ =& R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{3.7}$$

3.4 Unified Notation for Episodic and Continuing Tasks

3.5 Policies and Value Functions

1. **policy:** a mapping from states to probabilities of selecting each possible action. If the agent is following policy π at time t , then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

2. **state-value function for policy π:** the expected return when starting in s and following π thereafter:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] \quad \forall s \in \mathcal{S} \tag{3.8}$$

3. **action-value function for policy π:** the expected return starting from s , taking the action a , and thereafter following policy π:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{3.9}$$

4. **$v_\pi(s)$ vs $q_\pi(s,a)$:**

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) \tag{3.10}$$

5. **Bellman equation for v_π:** It states that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way. as show in Figure-3.1

$$\begin{aligned} v_\pi(s) \doteq& \mathbb{E}_\pi[G_t | S_t = s] \\ =& \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ =& \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\ =& \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{3.11}$$

6. **Bellman equation for q_π:**

$$\begin{aligned} q_\pi(s,a) =& \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma v_\pi(s')] \\ =& \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')] \end{aligned} \tag{3.12}$$

7. **$q_\pi(s,a)$ vs $v_\pi(s')$:**

$$q_\pi(s,a) = \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma v_\pi(s')] \tag{3.13}$$

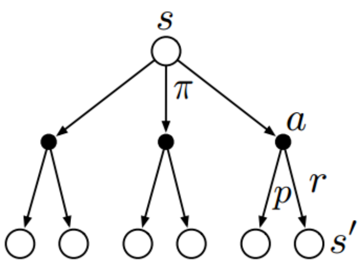

Figure 3.1: Backup diagram for v_π

3.6 Optimal Policies and Optimal Value Functions

1. **optimal policy:** A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}$ for all $s \in \mathcal{S}$. There is always at least one policy that is better than or equal to all other policies. This is an optimal policy π_*.

2. **optimal state-value function:**
$$v_*(s) \doteq \max_\pi v_\pi(s) \quad \forall s \in \mathcal{S} \tag{3.14}$$

3. **optimal action-value function:**
$$q_*(s,a) \doteq \max_\pi q_\pi(s,a) \quad \forall s \in \mathcal{S} \quad and \quad \forall a \in \mathcal{A}(s) \tag{3.15}$$

   For the state-action pair (s,a), this function gives the expected return for taking action a in state s and thereafter following an optimal policy.
$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] \tag{3.16}$$

4. **Bellman optimality equation for v_*:** Intuitively, the Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state. As show in Figure-3.2
$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t|S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \end{aligned} \tag{3.17}$$

5. **Bellman optimality equation for q_*:** As show in Figure-3.2
$$\begin{aligned} q_*(s,a) =& \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] \\ =& \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a')|S_t = s, A_t = a] \\ =& \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')] \end{aligned} \tag{3.18}$$

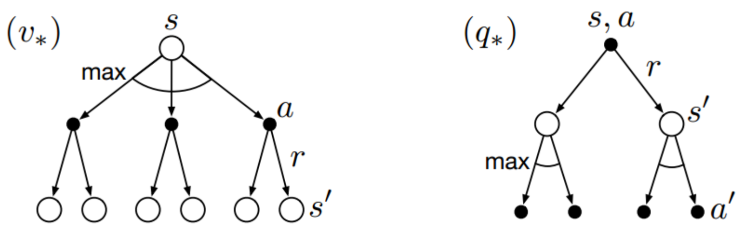

Figure 3.2: Backup diagram for v_* and q_*

Chapter 4

Dynamic Programming

The term dynamic programming (DP) refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision process (MDP). DP algorithms are obtained by turning Bellman equations such as these into assignments, that is, into update rules for improving approximations of the desired value functions.

4.1 Policy Evaluation (Prediction)

1. **policy evalution(prediction problem):** compute the state-value function $v_\pi$for an arbitrary policy π.

2. **iterative policy evaluation:** The existence and uniqueness of v_π are guaranteed as long as either $\gamma < 1$ or eventual termination is guaranteed from all states under the policy π.

$$v_\pi(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \tag{4.1}$$

   Update rule:

$$v_{k+1}(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')] \tag{4.2}$$

   Indeed, the sequence $\{v_k\}$ can be shown in general to converge to v_π as $k \to \infty$ under the same conditions that guarantee the existence of v_π.

3. **expected update:**All the updates done in DP algorithms are called expected updates because they are based on an expectation over all possible next states rather than on a sample next state.

4. **In-place iterative policy evaluation:** With two arrays for $v_k(s)$ and $v_{k+1}(s)$ respectively, the new values can be computed one by one from the old values without the old values being changed. Of course it is easier to use one array and update the values in place, that is, with each new value immediately overwriting the old one. show in Algorithm-4.1.

Algorithm 1 Iterative Policy Evaluation(In-place)

Require: the policy to be evaluated, a small threshold $\theta > 0$ determining accuracy of estimation, Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal)$=0.

```
1: repeat
2:     $\Delta \leftarrow 0$
3:     for $s \in \mathcal{S}$ do
4:         $v \leftarrow V(s)$
5:         $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
6:         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
7:     end for
8: until $\Delta < \theta$
```

4.2 Policy Improvement

1. **policy improvement theorem:** Let π and π' be any pair of deterministic policies such that, $\forall s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \tag{4.3}$$

   Then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in \mathcal{S}$.

$$v_{\pi'}(s) \geq v_\pi(s) \tag{4.4}$$

2. **policy improvement:** The process of making a new policy that improves on an original policy, by making it greedy with respect to the value function of the original policy. So far we have seen how, given a policy and its value function, we can easily evaluate a change in the policy at a single state to a particular action. It is a natural extension to consider changes at all states and to all possible actions, selecting at each state the action that appears best according to $q_\pi(s, a)$. In other words, to consider the new greedy policy, π', given by

$$\begin{aligned} \pi'(s) \doteq & arg \max_a q_\pi(s,a) \\ =& arg \max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ =& arg \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{4.5}$$

3. **optimality:** Suppose the new greedy policy, π', is as good as, but not better than, the old policy π. Then $v_\pi = v_{\pi'}$, and from 4.5 it follows that for all $s \in \mathcal{S}$:

$$\begin{aligned} v_{\pi'}(s) =& \max_a \mathbb{E}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) | S_t = s, A_t = a] \\ =& \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_{\pi'}(s')] \end{aligned} \tag{4.6}$$

   this is the same as the Bellman optimality equation 3.18, and $v_{\pi'}$ must be v_*, and both π and π' must be optimal policies. Policy improvement thus must give us a strictly better policy except when the original policy is already optimal.

4. **stochastic case:** the ideas of this deterministic case extend easily to stochastic policies.

4.3 Policy Iteration

Algorithm 2 Policy Iteration(using iterative evaluation) for estimating $\pi \approx \pi_*$

```
procedure INITIALIZATION
    $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily $\forall s \in \mathcal{S}$
end procedure

procedure POLICY EVALUATION
    repeat
        $\Delta \leftarrow 0$
        for $s \in \mathcal{S}($ do
            $v \leftarrow V(s))$
            $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
            $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
        end for
    until $\Delta < \theta$
end procedure

procedure POLICY IMPROVEMENT
    $policy\_stable \leftarrow true$
    for $s \in \mathcal{S}$ do
        $old_action \leftarrow \pi(s)$
        $\pi(s) \leftarrow arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
        if $old\_action \neq \pi(s)$ then
            $policy\_stable \leftarrow false$
        end if
    end for
    if $policy\_stable$ then
        stop and return $V \approx v_*$ and $\pi \approx \pi_*$
    else
        go to procedure POLICY EVALUATION
    end if
end procedure
```

4.4 Value Iteration

In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. One important special case is when policy evaluation is stopped after just one sweep (one update of each state).This algorithm is called value iteration, show in Algorithm-4.4. It can be written as a particularly simple update operation that combines the policy improvement and truncated policy evaluation steps:

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')] \end{aligned} \tag{4.7}$$

for all $s \in \mathcal{S}$. For arbitrary v_0 , the sequence v_k can be shown to converge to v_* under the same conditions that guarantee the existence of V_*.

Another way of understanding value iteration is by reference to the Bellman optimality equation 3.18, Also note how the value iteration update is identical to the policy evaluation update 4.2 except that it requires the maximum to be taken over all actions.

Algorithm 3 Value Iteration, for estimating $\pi \approx \pi_*$

```
Require: the policy to be evaluated, a small threshold $\theta > 0$ determining accuracy of estimation, Initialize $V(s)$, for all
    $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal)$=0.

1: repeat
2:     $\Delta \leftarrow 0$
3:     for $s \in \mathcal{S}$ do
4:         $v \leftarrow V(s)$
5:         $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
6:         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
7:     end for
8: until $\Delta < \theta$

9: Output a deterministic policy, $\pi \approx \pi_*$, such that
```

$$\pi(s) = arg \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

4.5 Asynchronous Dynamic Programming

A major drawback to the DP methods that we have discussed so far is that they involve operations over the entire state set of the MDP, that is, they require sweeps of the state set. **Asynchronous DP algorithms** are in-place iterative DP algorithms that are not organized in terms of systematic sweeps of the state set. If $0 \leq \gamma < 1$, asymptotic convergence to v_* is guaranteed given only that all states occur in the sequence $\{s_k\}$ an infinite number of times.

4.6 Generalized Policy Iteration

Letting policy-evaluation and policy improvement processes interact, independent of the granularity and other details of the two processes. That is, all have identifiable policies and value functions, with the policy always being improved with respect to the value function and the value function always being driven toward the value function for the policy. If both the evaluation process and the improvement process stabilize, that is, no longer produce changes, then the value function and policy must be optimal.

4.7 Efficiency of Dynamic Programming

DP methods take to find an optimal policy is polynomial in the number of states and actions.DP is sometimes thought to be of limited applicability because of the **curse of dimensionality**, the fact that the number of states often grows exponentially with the number of state variables. In fact, DP is comparatively better suited to handling large state spaces than competing methods such as direct search and linear programming.

Chapter 5

Monte Carlo Methods

5.1 Monte Carlo Prediction

1. **Monte Carlo methods:** Estimate value function(expected return) from experience, then, is simply to average the returns observed after visits to that state. As more returns are observed, the average should converge to the expected value.

2. **First-vist MC method vs every-visit MC method:** The first-visit MC method, as show in Algorithm-5.1, estimates $v_\pi(s)$ as the average of the returns following **first visits** to s , whereas the every-visit MC method averages the returns following **all visits** to s. Both first-visit MC and every-visit MC converge to $v_\pi(s)$ as the number of visits (or first visits) to s goes to infinity.

3. **Property of MC methods:**

   (a) The ability of Monte Carlo methods to work with sample episodes alone can be a significant advantage when one has no knowledge or even complete knowledge of the environments dynamics;

   (b) Monte Carlo methods **do not bootstrap**, the estimates for each state are independent. The estimate for one state does not build upon the estimate of any other state, as is the case in DP;

   (c) The computational expense of estimating the value of a single state is independent of the number of states.

Algorithm 4 First-visit MC prediction, for estimating $V \approx v_\pi$

Require: the policy π to be evaluated, Initialize $V(s)$, arbitarily, for all $s \in \mathcal{S}$, $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

```
1:  loop                                                        ▷ for each episode
2:      Generate an episode following $\pi$:$S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, A_{T-1}, R_T$
3:      $G \leftarrow 0$
4:      for each step of episode, $t = T-1, T-2, \dots, 0$ do
5:          $G \leftarrow \gamma G + R_{t+1}$
6:          if $S_t$ not in $S_0, S_1, \dots, S_{t-1}$ then        ▷ appear in episode for teh first time
7:              Append $G$ to $Returns(S_t)$
8:              $V(S_t) \leftarrow average(Returns(S_t))$
9:          end if
10:     end for
11: end loop
```

5.2 Monte Carlo Estimation of Action Values

1. **Necessity of estimation of action values:** Without a model, state values alone are not sufficient. One must explicitly estimate the value of each action in order for the values to be useful in suggesting a policy.

2. **First-visit vs ervery visit MC method:** The every-visit MC method estimates the value of a stateaction pair as the average of the returns that have followed all the visits to it. The first-visit MC method averages the returns following the first time in each episode that the state was visited and the action was selected.

3. **Maintainig exploration:** If $\pi$is a **deterministic policy**, then in following π one will observe returns only for one of the actions from each state. However, the purpose of learning action values is to help in choosing among the actions available in each state. To compare alternatives we need to estimate the value of all the actions from each state, not just the one we currently favor. For policy evaluation to work for action values, we must assure **continual exploration**.

4. **Exploring starts:** One way to do this is by specifying that the episodes start in a stateaction pair, and that every pair has a nonzero probability of being selected as the start. This guarantees that all stateaction pairs will be visited an infinite number of times in the limit of an infinite number of episodes.

5.3 Monte Carlo Control(Approximate Optimal Policies)

1. **Monte Carlo policy iteration:**In this method, we perform alternating complete steps of policy evaluation and policy improvement, beginning with an arbitrary policy π_0 and ending with the optimal policy and optimal action-value function:
$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} q_{\pi_*}$$
Policy improvement is done by making the policy greedy with respect to the current (action) value function.
$$\pi(s) \doteq arg\max_a q(s,a) \tag{5.1}$$
The policy improvement theorem then applies to π_k and π_{k+1} because, for all $s \in \mathcal{S}$,
$$\begin{aligned} q_{\pi_k}(s,\pi_{k+1}(s)) &= q_{\pi_k}(s, arg\max_a q_{\pi_k}(s,a) \\ &= max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s,\pi_k(s)) \\ &\geq v_{\pi_k}(s) \end{aligned} \tag{5.2}$$
the theorem assures us that each π_{k+1} is uniformly better than π_k , or just as good as π_k.

2. **Monte Carlo ES:**(Monte Carlo with Exploring Starts)

Algorithm 5 Monte Carlo ES

```
Require:
1: Initialize $\pi(s) \in \mathcal{A}(s)$(arbitarily), for all $s \in \mathcal{S}$,
2: Initialize $Q(s,a) \in \mathbb{R}$(arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$,
3: Initialize $Returns(s,a) \leftarrow emptylist$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$.

4: repeat
5:     Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all paris have probability $> 0$
6:     Generate an episode from $S_0, A_0$, following $\pi : S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$
7:     $G \leftarrow 0$
8:     for each step of episode, $t = T-1, T-2, ...., 0$ do
9:         $G \leftarrow \gamma G + R_{t+1}$
10:        if The pair $S_t, A_t$ not appears in $S_0, A_0, S_1, A_1, ..., S_{t-1}, A_{t-1}$ : then
11:            Append $G$ to $Returns(S_t, A_t)$
12:            $Q(S_t, A_t) \leftarrow average(Returns(S_t, A_t))$
13:            $\pi(S_t) \leftarrow arg\max_a Q(S_t, a)$
14:        end if
15:    end for
16: until Converge
```

5.4 Monte Carlo Control without Exploring Starts

1. $\epsilon - soft$ **policy:** meaning that $\pi a|s > 0$ for all $s \in \mathcal{S}$ and all $a \in \mathcal{A(f)}$, but gradually shifted closer and closer to deterministic optimal policy.

2. **On-policy first-visit MC control (for ϵ-soft policies), estimates $\pi \approx \pi_*$**

Algorithm 6 On-policy first-visit MC control

```
Require:
1:  Initialize $\pi \leftarrow$ an arbitrary $\epsilon - soft$ policy
2:  Initialize $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
3:  Initialize $Returns(s,a) \leftarrow emptylist$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

4:  repeat                                                         ▷ for each episode
5:      Generate an episode following $\pi : S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$
6:      $G \leftarrow 0$
7:      for each step of episode, $t = T-1, T-2, ...., 0$ do
8:          $G \leftarrow \gamma G + R_{t+1}$
9:          if The pair $S_t, A_t$ not appears in $S_0, A_0, S_1, A_1, ..., S_{t-1}, A_{t-1}$ : then
10:             Append $G$ to $Returns(S_t, A_t)$
11:             $Q(S_t, A_t) \leftarrow average(Returns(S_t, A_t))$
12:             $A^* \leftarrow arg \max_a Q(S_t, a)$
13:         end if
14:         for each $a \in \mathcal{A}(S_t)$ do
15:             if $a = A^*$ then
16:                 $\pi(a|S_t) \leftarrow 1 - \epsilon + \epsilon/|\mathcal{A}(S_t)|$
17:             else
18:                 $\pi(a|S_t) \leftarrow \epsilon/|\mathcal{A}(S_t)|$
19:             end if
20:         end for
21:     end for
22: until Converge
```

5.5 Off-policy Prediction via Importance Sampling

1. **off-policy learning:** Use two policies, one is the policy to be learned about and that becomes the optimal policy, called **target policy**, and one that is more exploratory and is used to generate behavior, called **behavior policy**.

2. **assumption of coverage:** Consider target policy π and behavior policy b, and both policies are fixed and given. In order to use episodes from b to estimate values for π, we require that every action taken under π is also taken under b. That is, we require that $\pi(a|s) > 0$ implies $b(a|s) > 0$. It follows from cob=verage that b must be stochastic in states where it is not identical to π.

3. **importance sampling:** a general technique for estimating expected values under one distribution given samples from another. We apply importance sampling to off-policy learning by weighting returns according to the relative probability of their trajectories occuring under target and behavior policies, called the **importance-sampling ratio**.

4. **importance ratio:** given a starting state S_t, the probability of subsequent state-action trajectories, $A_t, S_{t+1}, A_{t+1}, ..., S_T$, occuring under any policy π is

$$\begin{aligned} Pr\{A_t, S_{t+1}, A_{t+1}, ..., S_T | S_t, A_{t:T-1} \sim \pi\} &= \pi(A_t|S_t) p(S_{t+1}|S_t, A_t) \pi(A_{t+1}|S_{t+1}) ... p(S_T|S_{T-1}, A_{T-1}) \\ &= \prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k) \end{aligned} \tag{5.3}$$

the relative probability of the trajectory under the target and behavior policies (the importace sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} \tag{5.4}$$

5. **estimating the expected returns (values) under the target policy:** Get returns G_t due to the behavior policy. $v_b(s) = \mathbb{E}[G_t|S_t = s]$, the ratio $\rho_{t:T-1}$ transforms the $v_b(s)$ to $v_\pi(s)$.

$$\mathbb{E}[\rho_{t:T-1}G_t|S_t = s] = v_\pi(s) \tag{5.5}$$

6. **ordinary importance sampling:** To estimate $v_\pi(s)$, we simply scale the returns by the ratios and avrage the results

$$V(s) \doteq \frac{\sum_{t\in\mathcal{J}(s)} \rho_{t:T(t)-1}G_t}{|\mathcal{J}(s)|} \tag{5.6}$$

7. **weighted importance sampling:**

$$V(s) \doteq \frac{\sum_{t\in\mathcal{J}(s)} \rho_{t:T(t)-1}G_t}{\sum_{t\in\mathcal{J}(s)\rho_{t:T(t)-1}}} \tag{5.7}$$

8. **difference between the two importance sampling:** for the first-visit methods, ordinary importance sampling is unbiased whereas weighted importance sampling is baised (though the bias convergesasymptotically to zero).On the other hand, the variance of ordinary importance sampling is in general unbounded because the variance of the ratios can be unbounded, whereas in the weighted estimator the largest weight on any single return is one. In fact, assuming bounded returns, the variance of the weighted importance-sampling estimator converges to zero even if the variance of the ratios themselves is infinite. The every-visit methods for ordinary and weighed importance sampling are both biased, though, again, the bias falls asymptotically to zero as the number of samples increases.

Part II

Approximate Solution Methods

Bibliography