04_01_Classical Settings of Reinforcement Learning

This part almost has the same content as the 6 & 7 weeks' content of Artificial intelligence.

But some important and not overlapped part is illustrated here

1. Off-policy and On-policy

Introduction

Off-policy:

Use experience of different policies to improve π

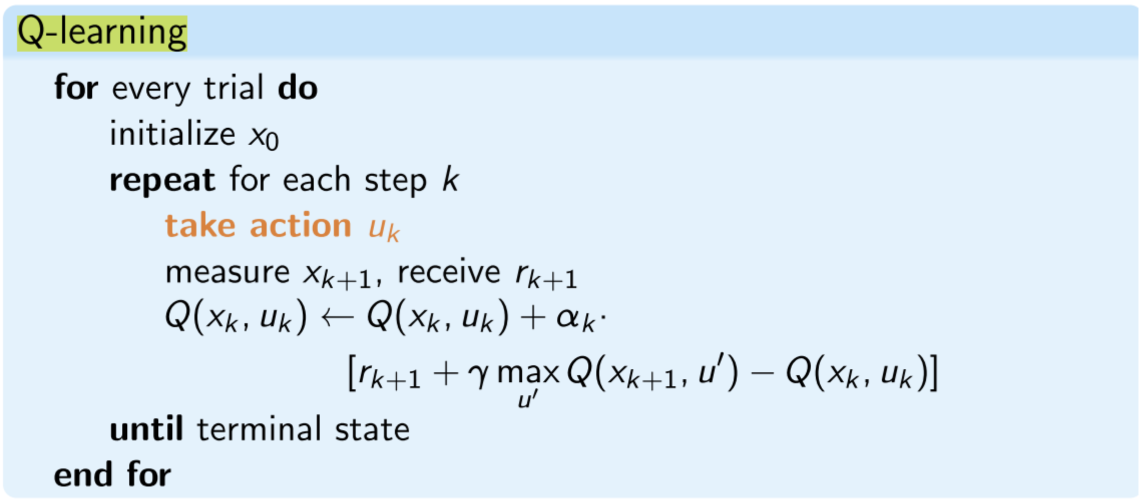
- find Q^*, use it to compute π^*
- find Q^π, improve π, repeat

On-policy:

Use experience of 1 to improve π

Q-learning

Online, off-policy



Q-learning

```
for every trial do
    initialize $x_0$
    repeat for each step $k$
        take action $u_k$
        measure $x_{k+1}$, receive $r_{k+1}$
        $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$
                $[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$
    until terminal state
end for
```

- Q-learning is called

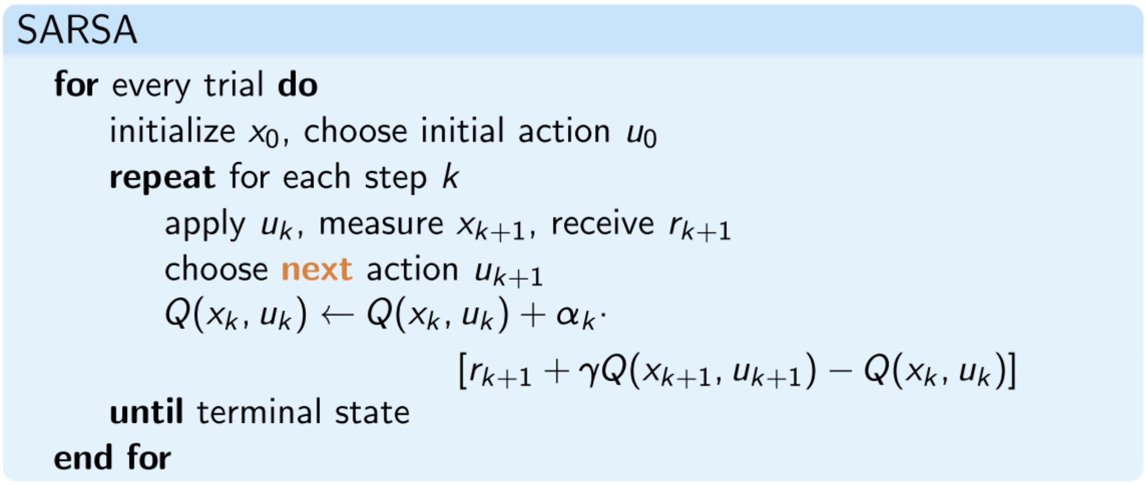
  off-policy

  because the updated policy is different from the behavior policy (use action take make value from x_{k+1} to update the $Q(x_k, u_k)$, while does not really take this action),

  - It estimates the reward for future actions and appends a value to the new state **without actually following any greedy policy.**

SARSA

online on-policy



SARSA

```
for every trial do
    initialize $x_0$, choose initial action $u_0$
    repeat for each step $k$
        apply $u_k$, measure $x_{k+1}$, receive $r_{k+1}$
        choose next action $u_{k+1}$
        $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$
                        $[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$
    until terminal state
end for
```

- SARSA (state-action-reward-state-action) is an

  on-policy

  reinforcement learning algorithm that estimates the value of the policy being followed.
    - In this algorithm, if the agent grasps the optimal policy and uses the same to act.

Exploration-Exploitation Trade-off: ϵ-greedy

$$u_k = \begin{cases} \arg\max_{\bar{u}} Q\left(x_k, \bar{u}\right) & \text{with probability } \left(1 - \varepsilon_k\right) \\ \text{a random action} & \text{with probability } \varepsilon_k \end{cases}$$

- For Q-learning, ϵ_k always decrease with time
- For SARSA, ϵ_k always decrease with time, and we always require $\lim_{k \to \infty} \varepsilon_k = 0$

2. Temporal Difference & Eligibility Trace

Temporal Difference

$$V\left(s_k\right) \leftarrow V\left(s_k\right)+\alpha_k\left[r_{k+1}+\gamma V\left(s_{k+1}\right)-V\left(s_k\right)\right],$$

Convergence

Convergence is guaranteed when α_k satisfies the conditions:

$$\begin{gathered}\textstyle\sum_{k=1}^{\infty} \alpha_k=\infty \\ \text{and } \textstyle\sum_{k=1}^{\infty} \alpha_k^2<\infty\end{gathered}$$

- The first condition is to ensure that the learning rate is large enough to overcome initial conditions or random fluctuations
- $V(s_k)$ is only updated after receiving the new reward r_{k+1}

Eligibility Trace

Background

Because $V(s_k)$ is only updated after receiving the new reward r_{k+1} in normal value iteration

- It is reasonable to say that immediate rewards should **change the values in all states that were visited** in order to get to the current state

Introduction

An extra variable, called the **eligibility trace** is associated with each state, indicating how eligible it is for a change when a new reinforcing event comes available.

$$\begin{gathered}\delta_k=r_{k+1}+\gamma V_k\left(s_{k+1}\right)-V_k\left(s_k\right) \\ \Delta V_k(s)=\alpha \delta_k e_k(s)\end{gathered}$$

Accumulating Traces

$$e_k(s)= \begin{cases}\gamma \lambda e_{k-1}(s) & \text{if } s \neq s_k \\ \gamma \lambda e_{k-1}(s)+1 & \text{if } s=s_k\end{cases}$$

Replacing Traces

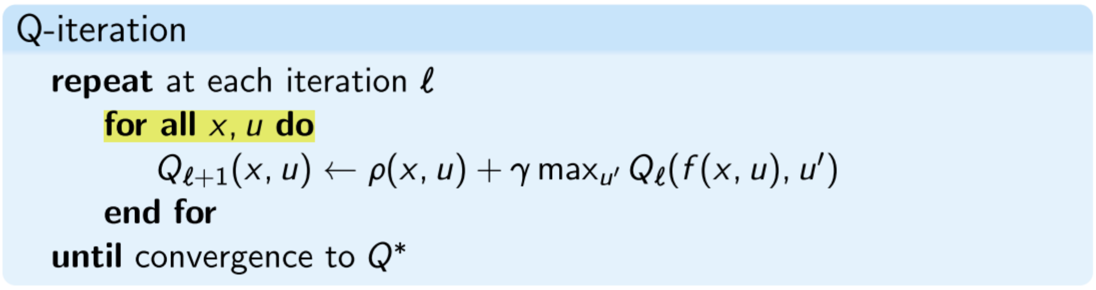
$$e_k(s) = \begin{cases} \gamma\lambda e_{k-1}(s) & \text{if } s \neq s_k \\ 1 & \text{if } s = s_k \end{cases}$$

- The method of accumulating traces is known to suffer from convergence problems, therefore **replacing traces is used almost always** in the literature to update the eligibility traces.
- Accumulating Traces update normal state with time-decreasing coefficient, while the second arrive at a state will have a larger than 1 overlapping effect,
- Replacing Traces do not has the problem in accumulatin traces

3. Q-iteration

Introduction

- Offline, Model-based solution
- Same as value iteration, but use Q-table



```
Q-iteration
  repeat at each iteration $\ell$
      for all $x, u$ do
          $Q_{\ell+1}(x,u) \leftarrow \rho(x,u) + \gamma \max_{u'} Q_\ell(f(x,u),u')$
      end for
  until convergence to $Q^*$
```

Convergence

Each update is a contraction with factor γ:

$$\|Q_{\ell+1} - Q^*\|_\infty \leq \gamma \|Q_\ell - Q^*\|_\infty$$

Q-iteration monotonically converges to Q^*

4. Initial Value Selection

> *From: http://incompleteideas.net/book/ebook/node21.html*

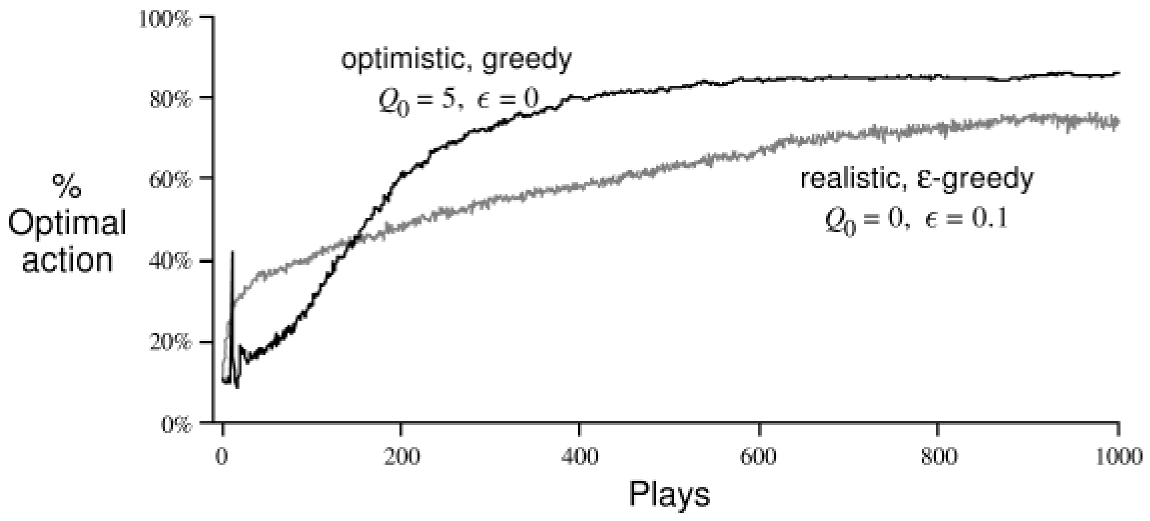
Optimistic Initial Values

All the methods we have discussed so far are dependent to some extent on the initial action-value estimates. In the language of statistics, these methods are biased

by their initial estimates. For the sample-average methods, the bias disappears once all actions have been selected at least once, but for methods with constant, the bias is permanent, though decreasing over time as given by (2.7)

In practice, this kind of bias is usually not a problem, and can sometimes be very helpful. 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 of encouraging exploration. Suppose that instead of setting the initial action values to zero, as we did in the 10-armed testbed, we set them all to +5. Recall that the

in this problem are selected from a normal distribution with mean 0 and variance 1. An initial estimate of +5 is thus wildly optimistic. But this optimism encourages action-value methods to explore. Whichever actions are initially selected, the reward is less than the starting estimates; the learner switches to other actions, being "disappointed" with the rewards it is receiving. The result is that all actions are tried several times before the value estimates converge. The system does a fair amount of exploration even if greedy actions are selected all the time.


% Optimal action
optimistic, greedy
$Q_0 = 5, \epsilon = 0$
realistic, ε-greedy
$Q_0 = 0, \epsilon = 0.1$
Plays