Reinforcement Learning: An Introduction - Chapter 8

Thomas Hopkins

Exercise 8.1

Table-lookup TD(λ) is a special case of general TD(λ) where each component of $\vec{\theta}_t$ represents a single state $s \in S$. The term $\nabla_{\vec{\theta}_t} V_t(s_t)$ becomes a 1-hot vector (zero vector with a 1 in a single component) that represents the state being updated. This method simply updates all of the states in one sweep using vector addition.

Exercise 8.2

Similar to the previous exercise, the state aggregation method has one component of the parameter vector per group of states. The gradient vector is simply a 1-hot vector representing which group of states, or component of the parameter vector, should be updated.

Exercise 8.3

Let

$$\Delta\vec{\theta}_t = \alpha[R_t^\lambda - V_t(s_t)]\nabla_{\vec{\theta}_t} V_t(s_t)$$

be the update for the on-line method (8.4). The off-line method would then have

$$\Delta\vec{\theta} = \alpha \sum_{t=0}^{T-1} [R_t^\lambda - V_t(s_t)]\nabla_{\vec{\theta}_t} V_t(s_t)$$

That is, the gradients need to be accumulated throughout the episode, then updated after the episode terminates.

Exercise 8.4

The off-line version of the backward view is

$$\vec{\theta} = \vec{\theta} + \alpha \sum_{t=0}^{T-1} \delta_t \vec{e}_t$$

where

$$\Delta\vec{\theta} = \alpha \sum_{t=0}^{T-1} \delta_t \vec{e}_t$$

Exercise 8.5

We can reproduce the tabular case of reinforcemnet learning using the linear framework by having a single feature per state. Then when that state is encountered, only that feature is turned on (given a value of 1) while all of the other features are turned off (given a value of 0). For most problems, this is not possible since there are far too many possible states.

Exercise 8.6

Similar to the previous exercise, we would have a single feature per group of states. As long as the state aggregation method is known beforehand, we can assign a single feature to a group and turn it on when a state in that group is encountered. All of the other features are turned off.

Exercise 8.7

A vertical or horizontal striped tiling would make sense here. The generalization occurs along the stripe while the discrimation occurs across it.

Exercise 8.8

The actor-critic control method can be extended to use function approximation by allowing the critic to estimate the value of a state using its own parameter vector while the actor uses its own parameter vector to estimate the policy.

Exercise 8.9

Optimal weights are the zero vector. The TD(0) method diverges in this case.

In [22]:

```
using Random

Random.seed!(32)
weights = [1.0 for i = 1:7]
weights[6] = 10.0
alpha = 0.01
gamma = 0.99

println("Starting weights: $weights")
for e = 1:5000
    # start state
    s = rand(1:5)
    q_s = weights[7] + 2 * weights[s]
    # next state is always 6
    sp = 6
    q_sp = 2 * weights[7] + weights[sp]
    delta = gamma * q_sp - q_s
    # derivative with respect to weight 7 is 1
    weights[7] = weights[7] + alpha * delta * 1
    # derivative with sepect to weight s is 2
    weights[s] = weights[s] + alpha * delta * 2
    # with probability 0.01 the episode ends
    # otherwise we repeat state 6
    while rand() > 0.01
        q = 2 * weights[7] + weights[6]
        delta = gamma * q - q
        weights[7] = weights[7] + alpha * delta * 2
        weights[6] = weights[6] + alpha * delta * 1
    end
    q = 2 * weights[7] + weights[6]
    delta = -q
    weights[7] = weights[7] + alpha * delta * 2
    weights[6] = weights[6] + alpha * delta * 1
end
println("After 100 episodes: $weights")
```

```
Starting weights: [1.0, 1.0, 1.0, 1.0, 1.0, 10.0, 1.0]
After 100 episodes: [1.720000000000135, 1.7200000000001403, 1.72000000000013
5, 1.7200000000001394, 1.720000000000135, 6.880000000000546, -3.4400000000002
71]
```