

Groundhog day

6. (15 points) We will be performing Q-learning in an MDP with states s_0 through s_5 , and actions a_1 and a_2 . Let the discount factor $\gamma = 0.8$ and learning rate $\alpha = 1$.

Note: do not assume anything more about the MDP from which this experience was generated. It is not necessarily the same as the one from question ??.

- (a) The Q-learning algorithm begins with $Q(s, a) = 0$ for all s and a . It receives the following experience, in order. Each tuple represents a state, action, reward, and next state.

$(s_0, a_2, 0, s_2)$

$(s_2, a_1, 0, s_3)$

$(s_3, a_1, 0, s_1)$

$(s_1, a_1, 10, s_0)$

$(s_0, a_1, 0, s_5)$

$(s_5, a_1, 0, s_4)$

$(s_4, a_1, 5, s_0)$

Fill in the resulting Q values in the following table:

	s_0	s_1	s_2	s_3	s_4	s_5
a_1						
a_2						

- (b) Iyaz suggests that, rather than getting new experience, it would be a good idea to replay this data over several times using the regular Q-learning update. What's the minimum number of times you would have to iterate through this data before $Q(s_0, a_2) > Q(s_0, a_1)$?

Note: it should be possible to answer this question by thinking about the structure of the problem, rather than by grinding through more Q-learning update calculations.

Name: _____

(c) Now, imagine that you have a very large memory full of experience that explores the whole environment effectively. In the following questions, you can assume that a neural network `nn` has the following methods:

- `nn.train(data, epochs)` takes a list of (x, y) pairs and does supervised training on it for the specified number of epochs. If `epochs` is `None` it trains until convergence.
- `nn.predict(x)` takes an input x and returns the predicted y value
- `nn.init()` randomly reassigns all the weights in the network.

Consider the following code.

```
# nn = dictionary of neural networks, one for each action
# each nn[a] maps state s into Q(s, a)
gamma = 0.8
for t in range(max_iterations):
    for a in actions:
        data[a] = [(s, <fill in>) for (s, a_prime, r, s_prime) in memory
                    if a_prime == a]

    for a in actions:
        nn[a].train(data[a], <epochs>)
```

What is a correct expression for `<fill in>` above?

Don't panic about detailed Python syntax; you may use words to clarify if you think you might be wrong.

What is an appropriate value for `<epochs>` above?

Name: _____

(d) If we change the loop to have the form

```
for t in range(max_iterations):  
    for (s, a, r, s') in memory:  
        data = [(s, <fill in>)] # a single data point  
        nn[a].train(data, <epochs>)
```

Provide a value for <epochs> above that will cause this algorithm to converge to a correct solution or explain why no such value exists.

(e) Would it be okay to call `nn[a].init()` on the line before calling `train` in the code loop in part ???

Yes No

Provide a short explanation of your answer.

Name: _____

(f) Would it be okay to call `nn.init()` on the line before calling `train` in the code loop in part ???

Yes No

Provide a short explanation of your answer.

(g) We often use ϵ -greedy exploration in Q learning, in which we execute the action with the highest Q value in the current state with probability $1 - \epsilon$ and execute a random action with probability ϵ . What problem might occur if we set ϵ to be too small?