Exercises Reinforcement Learning: An Introduction

Contents

Code for exercises can be found at github.com/brynhayder

1 Chapter 1

1.1 Exercise 1.1: Self-Play

Q

Suppose, instead of playing against a random opponent, the reinforcement learning algorithm described above played against itself, with both sides learning. What do you think would happen in this case? Would it learn a different policy for selecting moves?

A

- Would learn a different policy than playing a fixed opponent since the opponent would also be changing in this case.
- May not be able to learn an optimal strategy as the opponent keeps changing also.
- Could get stuck in loops.
- Policy could remain static since on average they would draw each iteration.

1.2 Exercise 1.2: Symmetries

Q

Many tic-tac-toe positions appear different but are really the same because of symmetries. How might we amend the learning process described above to take advantage of this? In what ways would this change improve the learning process? Now think again. Suppose the opponent did not take advantage of symmetries. In that case, should we? Is it true, then, that symmetrically equivalent positions should necessarily have the same value?

A

- We could label the states as unique up to symmetries so that our search space is smaller, this way we will get a better estimate of optimal play.
- If we are playing an opponent who does not take symmetries into account when they are playing then we should not label the states as the same since the opponent is part of the environment and the environment is not the same in those states.

1.3 Exercise 1.3: Greedy Play

Q

Suppose the reinforcement learning player was greedy, that is, it always played the move that brought it to the position that it rated the best. Might it learn to play better, or worse, than a nongreedy player? What problems might occur

A

- The greedy player will not explore, so will in general perform worse than the non-greedy player
- If the greedy player had a perfect estimate of the value of states then this would be fine.

1.4 Exercise 1.4: Learning from Exploration

Q

Suppose learning updates occurred after all moves, including exploratory moves. If the step-size parameter is appropriately reduced over time (but not the tendency to explore), then the state values would converge to a set of probabilities. What are the two sets of probabilities computed when we do, and when we do not, learn from exploratory moves? Assuming that we do continue to make exploratory moves, which set of probabilities might be better to learn? Which would result in more wins?

A

I think that an estimate for the probability of the state producing a win should be based on the optimal moves from that state.

- The one in which we only record the optimal moves is the probability of our optimal agent winning. If we include exploration then this is the probability of the training agent winning.
- Better to learn the probability of winning with no exploration since this is how the agent will perform in real time play.
- Updating from optimal moves only will increase probability of winning.

1.5 Exercise 1.5: Other Improvements

Q

Can you think of other ways to improve the reinforcement learning player? Can you think of any better way to solve the tic-tac-toe problem as posed?

A

I'm not too sure here...

- We could rank the draws as better than the losses.
- We might like to try running multiple iterations of games before updating our weights as this might give a better estimate.

2 Chapter 2

2.1 Exercise 2.1

Q

In ε-greedy action selection, for the case of two actions and $\varepsilon = 0.5$, what is the probability that the greedy action is selected?

A

0.5.

2.2 Exercise 2.2: Bandit example

Q

Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ε-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1$, $R_1 = 1$, $A_2 = 2$, $R_2 = 1$, $A_3 = 2$, $R_3 = 2$, $A_4 = 2$, $R_4 = 2$, $A_5 = 3$, $R_5 = 0$. On some of these time steps the ε case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

A

A_2 and A_5 were definitely exploratory. Any of the others *could* have been exploratory.

2.3 Exercise 2.3

Q

In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

A

The $\varepsilon = 0.01$ will perform better because in both cases as $t \to \infty$ we have $Q_t \to q_*$. The total reward and probability of choosing the optimal action will therefore be 10 times larger in this case than for $\varepsilon = 0.1$.

2.4 Exercise 2.4

Q

If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters?

A

$$Q_{n+1} = \alpha_n R_n + \sum_{i=1}^{n-1} \prod_{k=0}^{i} (1 - \alpha_{n-k}) \alpha_{n-i} R_{n-i} \tag{1}$$

2.5 Exercise 2.5 (programming)

Q

Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for non-stationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (say by adding a normally distributed increment with mean zero and standard deviation 0.01 to all the $q_*(a)$ on each step). Prepare plots like Figure 2.2 for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\varepsilon = 0.1$ and longer runs, say of 10,000 steps.

A

This is a programming exercise. For the relevant code/notebook please see the repo.