RL Textbook Exercises

Ryan Rudes

December 27, 2020

1 Chapter 1: Introduction

1.5 An Extended Example: Tic-Tac-Toe

Exercise 1.1: Self-Play 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?

The algorithm will continue to learn, converging to optimal values. It is possible that it will even reach a higher skill level because it may pursue unique strategies which may lead it to discover more optimal state evaluations due to better exploration.

Exercise 1.2: Symmetries 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?

If we encounter a state with symmetries, we should reflect over those symmetries to map to a simpler state. This will reduce the state space and reduce the complexity of the task, so yes, we should exploit symmetrically equivalent positions. However, symmetrically equivalent positions don't necessarily hold the same value in a multi-player game. The key reason is the fact that in a *multi*-player game, there is an opponent, and the state value depends on not only the agent's actions but also the actions taken by its opponent. For example, if the agent's opponent struggled to evaluate a particular orientation of a symmetrical position, but could successfully evaluate a different orientation of the essentially identical position, then the expected outcome of the game would differ depending upon which orientation

of the symmetrically equivalent positions was actually encountered in gameplay.

Exercise 1.3: Greedy Play 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?

It will likely learn to play less optimally than a more exploratory agent. If the agent is always forced to play what appears to be the most optimal action at each timestep, it will continuously repeat similar trajectories, and won't discover many new strategies in this repetitive gameplay. For this reason, a more exploratory agent will often outperform a greedy agent.

Exercise 1.4: Learning from Exploration 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 different set of probabilities. What (conceptually) 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?

The set of probabilities resulting from when we *do* learn from exploratory moves is the expected outcome from each state when taking a combination of optimal actions and random actions in accordance with the exploration strategy employed. The resulting set of probabilities when we do *not* learn from exploratory moves is the expected outcome from each state of exclusively taking the optimal actions according to the current state evaluations. It is better to exclude exploratory moves from our state updates because if the agent perhaps takes an arbitrary move when exploring from a particular state, and this action transitions to a poor outcome, it does not necessarily imply that the state from which the arbitrary action was taken was low-value, it may that the randomly-selection action was simply a poor choice. Therefore, learning from exploratory moves can make the optimization problem more random, which could hinder appropriate state evaluations for states in which exploratory actions were taken.

Exercise 1.5: Other Improvements 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?

- To improve convergence, we could reduce the impact of older updates, which become more irrelevant as more appropriate evaluations emerge in later stages
- Make the reduction of the learning rate parameter α proportional to the variance of the opponent's actions

2 Chapter 2: Multi-armed Bandits

2.2 Action-value Methods

Exercise 2.1 In ϵ-greedy action selection, for the case of two actions and $\epsilon = 0.5$, what is the probability that the greedy action is selected?

The probability of selecting the greedy action is $0.5 + 0.5 \cdot 0.5 = 0.75$**. There is a 50% probability that a random action will be selected. There are 2 actions, and thus there is a 25% probability that the greedy action will be selected as a random action. There is also the other 50% probability that the greedy action is intentionally selected.**

2.3 The 10-armed Testbed

Exercise 2.2: Bandit example 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?

In order to determine whether a greedy action or random action was selected at each timestep, we should tabulate the Q values for each action at each timestep. If the selected action at a particular timestep t did not have the maximum Q value at that timestep, then it was an exploratory action.

	a			
t	1	2	3	4
1	+0.00	+0.00	+0.00	+0.00
2	−1.00	+0.00	+0.00	+0.00
3	−1.00	+1.00	+0.00	+0.00
4	−1.00	−0.50	+0.00	+0.00
5	−1.00	+0.33	+0.00	+0.00

Table 1: Action-values at each timestep t

As we can see, the optimal actions each timestep t are as follows:

t=1 At initialization, all actions have the maximum value of 0 (action 1 was selected)

t=2 Actions 2-4 all possess the maximum value (action 2 was selected)

t=3 Action 2 is optimal (action 2 was selected)

t=4 Actions 3 and 4 both possess the maximum value (action 2 was selected)

t=5 Action 2 is optimal (action 3 was selected)

According to the optimal actions at each timestep, and the actions that were actually selected, we can deduce the following:

t=1 Undetermined

t=2 Undetermined

t=3 Undetermined

t=4 Definitely **exploratory** action because selected action is non-optimal

t=5 Definitely **exploratory** action because selected action is non-optimal

Exercise 2.3 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.

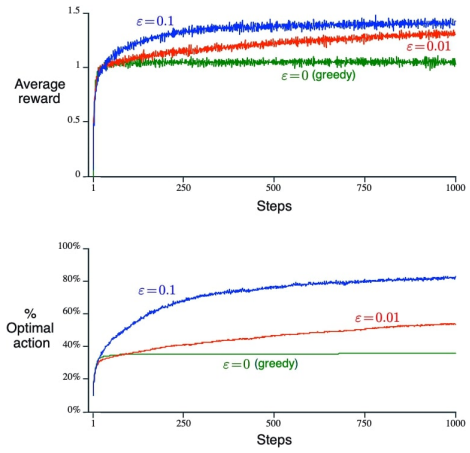

Figure 2.2: Average performance of ϵ-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

The best action is selected with probability $(1-\epsilon)+\epsilon * \frac{1}{k}$**. With** $\epsilon = 0.01$**, this is equal to a probability of** $0.99 + 0.01 \cdot 0.1 = 0.991$**. With** $\epsilon = 0.1$**, this is equal to a probability of** $0.9+0.1 \cdot 0.1 = 0.91$**. Therefore, in the long-run,** $\epsilon = 0.01$ **will converge to a superior optima than** $\epsilon = 0.1$**.**

2.5 Tracking a Nonstationary Problem

Exercise 2.4 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?

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\big[R_n - Q_n\big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\big[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\big] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2\alpha R_{n-2} + \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned} \tag{2.6}$$

Here's my solution for the weighting of each prior reward for the general case, in terms of the sequence $\{\alpha_i\}_{i=1}^n$

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha_n\big[R_n - Q_n\big] \\
&= \alpha_n R_n + (1-\alpha_n)Q_n \\
&= \alpha_n R_n + (1-\alpha_n)\big[\alpha_{n-1} R_{n-1} + (1-\alpha_{n-1})Q_{n-1}\big] \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1} \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\big[\alpha_{n-2} R_{n-2} + (1-\alpha_{n-2})Q_{n-2}\big] \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\alpha_{n-2} R_{n-2} + (1-\alpha_n)(1-\alpha_{n-1})(1-\alpha_{n-2})Q_{n-2} \\
&= \alpha_n R_n + Q_1 \prod_{i=1}^{n}(1-a_i) + \sum_{i=1}^{n-1}\Big[a_i R_i \prod_{j=i+1}^{n}(1-\alpha_j)\Big]
\end{aligned}$$