Exercise Guide for *Reinforcement Learning: An Introduction* by Sutton and Barto

Muthu Chidambaram

Last Updated: July 21, 2019

Contents

About

"Machines take me by surprise with great frequency." - Alan Turing

These notes contain my solutions to exercises from the book *Reinforcement Learning: An Introduction* by Richard S. Sutton and Andrew G. Barto.

1 Introduction

1.1 Exercise 1

It seems intuitive that the described self-play would converge to a maximin (or minimax, depending on order of play) policy, since the agent is competing with itself (alternating between minimizing and maximizing its value function).

We can make this notion more precise by introducing some notation. Suppose that we are playing Tic-Tac-Toe with "X" going first. Let s be an arbitrary game state, $V(s)$ be the probability that "O" wins from s, $N(s)$ be the neighboring states of s, and $A_X(s)$ and $A_O(s)$ be the next states chosen by the two respective player policies. Then we have

$$\begin{aligned} A_X(s) &= \arg\max_{s' \in N(s)} 1 - V(s') = \arg\min_{s' \in N(s)} V(s') \\ A_O(s) &= \arg\max_{s' \in N(s)} V(s') \\ V(s) &\leftarrow V(s) + \alpha[V(A_O(A_X(s))) - V(s)] \end{aligned}$$

from which we can see that if $V(s)$ converges, it converges to a maximin policy. This policy is not necessarily the same as the one learned by playing against a random opponent for the reasons stated in the text (a random opponent could make suboptimal moves).

1.2 Exercise 2

By mapping rotations and reflections of a given game state to a single state, we could significantly shrink the size of our value function table along with the search space of moves. However, we can only do this if the opponent also regards symmetric game positions as identical. If the opponent takes different actions in symmetric positions, then treating symmetric positions as the same would lead to sub-optimal play on our part.

1.3 Exercise 3

A greedy player could converge to a local optima, which may be worse than a nongreedy player who explores more of the search space. To illustrate the kind of problems that could occur, consider the following simplified scenario: the greedy player has just made its best move from position s, and the opponent must choose between two moves leading to states u and v respectively. Suppose we only have one move from either of the positions u and v, and that we always lose from u and always win from v. Furthermore, suppose the opponent chooses the move that leads to v with probability $p \gg V(s)$. Then, letting s' be the final state we reach, we can compute the change to $V(s)$ as

$$\mathrm{E}[\alpha(V(s') - V(s))] = \alpha(\mathrm{E}[V(s')] - V(s)) = \alpha(p - V(s)) > 0$$

so we will most likely keep making the same move from s. Thus, if there is another move from s that would actually lead to a certain win against the opponent's policy, we will not discover it.

1.4 Exercise 4

When we do not learn from exploratory moves, our expected updates are as described in the text:

$$V(s) \leftarrow V(s) + \alpha[V(s') - V(s)]$$

with $V(s)$ indicating the probability that we win from state s assuming we make our best moves. If we instead explore with likelihood ϵ, then our expected updates look more like:

$$V(s) \leftarrow V(s) + \alpha[(1-\epsilon)V(s') + \epsilon \sum_{s'' \neq s'} P(s'')V(s'') - V(s)]$$

where $P(s'')$ indicates the probability with which we end up in the non-greedy state s'' during exploration. In this case, we can interpret the probability $V(s)$ to be the expected probability of winning assuming we make our best move with probability ϵ and an exploratory move with probability $1 - \epsilon$.

If we assume that we want to keep making exploratory moves, then it makes more sense to learn the latter set of probabilities, as they incorporate information from all possible moves from a given state. In terms of winning, however, it seems like the former set of probabilities would be better, as they hone in on the probabilities corresponding to the best possible outcomes.

1.5 Exercise 5

Some potential improvements that come to mind:

- Track whether opponent plays symmetrically or not during learning, and then condense value function table afterwards if possible.
- If at any point during learning we play a sequence of moves that leads to a win, try to play the exact same sequence of moves again. Keep doing so for as long as it is possible to play the same sequence. This would help against a deterministic opponent - if we can win once, we can always win.
- We can prune the search space. We can disregard (set winning probability to 0) all states from which the opponent only has a single move, with that move being a winning move.

Perhaps a better approach to this problem would be to combine reinforcement learning with minimax optimal play. In other words, if we reach a state where we can win no matter what the opponent does, then there is no need to learn anything; we just use the minimax strategy from that state.