Exercises - Chapter 1

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?

My answer:

In the early stages of training, I believe it's likely that who the winning player is will switch back and forth frequently. Let's assume that player X wins the first game. Then player O will reduce the value of the state its last move took it to and will not select that move in the next game unless selected as an exploratory move.

In the long run, I think both players will learn a policy that can't lose except if an exploratory move is taken. All games will result in a draw, except when one player makes a losing exploratory move - which the other player will have learned to capitalize on and win.

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?

My answer:

We could take advantage of the symmetries by representing multiple states as one by combining the value function for these states. Updating the value for one state would update the value for all of its symmetrical states as well. This would improve the learning process by making it more efficient, the symmetrical states would combine what was learned and the agent wouldn't have to relearn the same thing in each state.

If the opponent doesn't take advantage of symmetries, we shouldn't either. Symmetrically equivalent positions shouldn't necessarily have the same value. If the opponent plays them differently in way that changes our chances of winning, they should have different values. In a more complex problem, for example when we are trying to generalize to a population of opponents, it might be better to take advantage of symmetries even if our opponents don't.

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?

My answer:

It might learn to play better, worse, or equally good, depending on the nongreedy player and randomness. The problem is that a greedy player can get stuck with selecting the same moves over and over. It's unlikely to try and learn from all possible moves and is thus likely to miss out on learning the best policy. It's still possible for the greedy player to learn to play better due to luck or if the nongreedy player explores too much.

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?

My answer:

When we learn from exploratory moves, the state values would converge to the probablities of winning with exploratory moves as a part of the policy (the actual probablities of winning from each state in this scenario). When we don't learn from exploratory moves, the state values would converge to the probabilites of winning from each state given optimal play by our player.

If we continue to make exploratory moves, it's probably best to learn the former probablities. It should result in equally many or more wins due to the agent selecting moves (when not exploring) using the actual probabilites of winning from each state. A state that is the best to get to given optimal play might not be the best when we are sometimes exploring.

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?

My answer:

One way to improve the player would be to give it more information about the game. We can figure out all states from where our player can force a win. For example, if we play the first move to the middle square and the opponent doesn't respond with a corner-move we can force a win. We could give all those objectively winning states value 1 and hardcode the policy to play the winning moves from them. Assuming that the opponent is playing a fixed policy, we should also reduce the exploration probability over time, eventually reaching 0.

Again assuming that the opponent is playing a fixed policy, a better way to solve the tic-tac-toe problem might be to try to reach all states quickly, except the ones where our player can force a win regardless of the opponent's policy, and learn the opponent's policy. We can then use our knowledge about the game to create a fully deterministic policy that never explores, and wins whenever possible and draws otherwise.