Chapter 1 Exercises

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?

In the Tic-Tac-Toe example, the opponent was an imperfect player, meaning the RL policy would converge to learning the most optimal moves against the player which may exploit some faults of the player. If the opponent was also a perfect player, then the policy would converge to also playing the most optimal moves resulting in always a tie. Similar in self play, as the policy continues to learn, it will converge to an optimal policy and it will always result in a tie. In a sense, it will be a different policy compared to the one playing against an imperfect opponent.

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?

We can amend the learning process to maintain one probability or value for all the symmetric states through rotation or reflection and when one of these are updated, essentially the probabilities for all symmetric states will be updated. This will most likely speed up learning and make the policy more "data efficient" since it can update the value of all corresponding states with just one sample. If the opponent, for example, would always lose if the policy put an X on the top left, then in that case we should not be considering symmetric states as equivalent. In the asymmetric policy, the state with an X in the top left would have extremely high value, while in the symmetric policy, these will average out with the other corners and won't indicate a big advantage.

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?

A greedy player may not explore all of the state space and converge to some local optima that it believes is best against an opponent. For example, if the opponent constantly lost to a particular set of moves, the policy will converge to have very high value associated with those sequence of moves and thus will constantly play them. However, if the opponent suddenly began to switch up moves, it will attempt to greedily choose from states that it has not seen or does not have enough data for, and will not play optimally.

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?

1. If we learn from exploratory moves, then probability is the probability of winning when acting according to a policy that takes random moves with a certain probability ϵ
2. If we don't explore at all, then probability is the probability of winning when acting greedily according to a policy

If we continue to make exploratory moves, then it makes sense for our probability of winning to also take into consideration the random actions that our policy and will then also result in more wins. The greedy probability doesn't take into consideration the exploratory moves at all, which may or may not play the greedy move.

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?

For more complex tasks, we can use a neural network to identify a mapping between the state space or the subspace of the state space to certain actions. The value function can also be conditioned on (s, a) pairs rather than just s to provide additional information that it can make the decisions from. As stated before, we can have some exploratory probability that is reduced over time as the policy learns.