My attempts to exercise in Reinforcement Learning book Chapter 1

Mengliao Wang

June 14, 2017

Exercise 1.1: *Self-Play*

The algorithm will still converge to the optimal policy, and possibly with a better performance and with less games to reach it. The idea is more conceptual without mathematical proof: the alrogithm does not depend on how the opponent plays, but only on the possible states opponent provided. Thus when playing against itself, at the beginning it will visit more sub-optimal moves, but in the end as the opponent improves, the policy will drop on the best moves much more easily and quickly. Eventually the converged probabilities given by the same optimized algorithm should be better than the ones given by a random opponent.

However my guess is there might be a need to increase the exploration, especially when these two players have similar initialization values, since the mutal coverage of optimal move selection will be much smaller, which cause it harder to explore other possible moves.

Exercise 1.2: *Symmetries*

At the beginning when we initiate the states, we should remove the "duplicate" states, which are states with the same positions after rotating or flipping the board. This will reduce the number of states in our search space significantly, thus speeding up the learning speed.

Yes we should. Even without remove these symmetries from the states list, they will eventually converge to the same value, maybe at different time depending on which one is explored more frequently.

Exercise 1.3: *Greedy Play*

It might play better at the very beginning than a non-greedy player, but the performance will drop quickly and eventually be much worse than a non-greedy player. The problem is that by totally discarding exploration, some potential optimal move might never be visited,

especially when their initial values are assigned too low.

Exercise 1.4: *Learning from Exploration*

When we do not learn from exploratory moves, the probability set is the chance of winning that gets esimated by updating the value backward. The learning process is passing the winning chance estimate from the final states (the only ones with reward - either 0 or 1) slowly back to the middle states, given this is the best move by far. If we update the value for exploration moves as well, then a wrong state will get updated and the whole chain will be updated incorrectly. For example, let's assume that state a has three possible moves to b_1, b_2, b_3. Here b_1 will lead to a winning condition after one move, thus has the highest value (close to 1) that is updated by previous runs. Also b_3 will lead to a losing condition after one move, thus has the lowest value (close to 0) that is updated previously. If we pick b_3 as an exploratory move and we learn from it, then state a will be updated to closer to 0, while ideally it should be closer to 1. So the set of probabilities when we do not learn from exploratory moves would be better, and results in more wins.

Exercise 1.5: *Other Improvements*

Some ideas that I have to improve this algorithm:

- We can add some human expert knowledge as prior to the initial values, instead of using 0.5 everywhere. This can help the algorithm converge faster and not miss some potential optimal states.

- Given the size of states search set (less than $3^9 \approx 20000$, and more duplicate/impossible states can be removed), the training will negligible from time perspective, so we can introduce more exploration than exploitation during the training process.

- This algorithm does not treat the fact of who plays first differently, which is a very important fact in tic-tac-toe game. However I don't think there is a need to separate the training process into two. We just need to make sure during training that each palyer has about the same numbers of games played first. Otherwise for example, if we keep training on the situation that our opponents starts first, almost half of the states will never be visited thus not trained at all.