Exercises Lecture 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?

Answer
It would learn a completely different policy, a pretty decent one, too. Not only would it improve the play strength as opposed to when learning against a fixed opponent, but it would also further improve the power of play due to the reactions against its own moves.

A good example of such an application is a chess engine. It plays against itself and updates weights in a different pattern, optimising the strategic approach to its actions and maximising the rewards.

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?

Answer
We can take advantage of 4 axes of symmetry to reduce the size of the board by 75%. Such an optimization would improve inference and reduce memory requirements. Should the opponent not take advantage of symmetries, it could then result in a worse overall performance.
For instance if the opponent misplays in one of the corners, then using symmetry-based gameplay would mean you wouldn't take advantage of the misplay. Therefore, when playing against an opponent, symmetrically equivalent positions always hold the same value.

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 non greedy player? What problems might occur?

It is going to play worse than a non greedy player overall. The more actions there are, the less chance of hitting that positive reward (greedy) action. In the long run it gets even worse. The greedy player loses all the versatility and would suffer from even the slightest variations in gameplay of the opponents.

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?

Answer
Given the step-size over time reduction, and a fixed exploration chance, the agent that doesn’t learn from exploratory moves has a probability set that is equal to the value of each state given the optimal action.
If the agent does learn from exploration, its probability set is equal to the expected value of each state as well as the exploration policy.
Not learning from exploration yields better results, as it reduces variance from suboptimal future states. Such an agent would win more if the rest of the parameters are equal.

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?

Answer
Weight decay would allow an agent to converge faster and better. Calibrating the exploration rate and learning based on the variance in the opponent's actions. If the variance is high, increase epsilon. If the opponent makes moves and you are constantly winning, reduce epsilon as much as possible (up to a complete greedy mode). Assessing an opponent in tic-tac-toe is key.