1 Introduction

We are considering a tic-tac-toe playing reinforcement learning agent.

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?

- When self-playing, the reinforcement agent would optimize for states where X wins, while the opponent would optimize for states where O wins.
- It is interesting that draws and losses are equally valued here. This means that the agent "won't mind" losing. It only values winning higher than losing and drawing. Of course, in self play this property is symetrically present. m
- While it seems intuitive that a sequence of games (given enough learning time) would always converge to a series of draws, given the specified reward function, it seems plausible that we may still see wins on both sides.
- Since the opponent keeps changing, it is not certain that the agent will learn an optimal strategy

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 program the states so that symmetric positions appear identical. For example, we may rotate them to uniquely maximize a certain sum of coefficients. This would speed up the learning process since there is less redundant information to be learned.
- If the opponent treats symmetrically equivalent positions differently, then our play might take that into account. Then, we do not necessarily need to treat them as equal.

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 best. Might it learn to play better, or worse, than a nongreedy player? What problems might occur?

- Such a player would not do exploration. It might get stuck in a local optimum.

1.4 Learning from Exploration

Suppose learning updates occured 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?

- The issue here is that the moves prior to the exploratory move would receive a weight update that is based on exploratory play. If I play three perfect moves, but then blunder the game because of an exploratory move, the three perfect moves would receive a negative update. We might get away with this if the tendency to explore is low enough, but still, basing results on optimal play seems liuke the better option.

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?

- We should rank draws higher than losses (but lower than wins)