Reinforcement learning (CS6700)
Written assignment #1

Aravind S
EE14B013

21st Jan. 2017

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?

If the agent plays against itself, for a given board position the state as seen by the 2 agents is different. So when the agent 1 encounters a WIN (reward of 1), agent 2 encounters a LOSE (reward of 0) and hence corresponding values are updated. As more moves are made, more updates are done and eventually the agent learns the optimal value function. And the corresponding policy is also learnt. On the other hand, if there are lot of DRAW positions when compared to WIN/LOSE while learning, the agent effectively learns to DRAW also. Note that WIN/LOSE are similar here. This is becaues WIN for agent 1 is LOSE for agent 2. WIN for agent 2 is LOSE for agent 1. But DRAW is same for both. So, if more DRAW positions are encountered and if some positive reward (0.5 for DRAW) is given for it, the agent captures that representation also. So, given a board position where agent 1 can either LOSE or DRAW => agent 2 can either WIN or DRAW, agent 1 will make sure it ends up in a DRAW position. This is because the agent would have learnt how to effectively DRAW a game not just WIN it. We encounter this type of situation here because the agent is playing with itself.

So, the agent learns from its moves and becomes better after each update. Eventually it learns optimal behaviour. The type of moves it makes can be altered by modifying rewards accordingly.

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?

For solving a reinforcement learning problem, we need to identify what is a state, action and reward. Reward is straightforward in this case. Say 1 for winning, 0 in case of losing/draw. Since, it is a 2-player game actions refer to putting an 'X' or an 'O' in a blank space. State, on the other hand can have different possible definitions. It is some encoding of the current board position. Symmetry can be exploited in obtaining a more compact state space representation.

Corresponding to the new definition of state, actions should also be encoded in a different way. Thus, exploiting symmetry we can get a more compact representation of state space and action set. It reduces the number of states we operate on and hence simplifies the problem.

When the opponent is taking advantage of symmetry, our algorithm should also exploit symmetry. It enables us to be a better player against this kind of opponent. On the other hand, if the opponent was not taking advantage of symmetry, we should not. It will decrease our performance on when playing with this type of opponent because we are enforcing symmetry on the opponent which is not there. For instance, if the opponent makes a right move from one position but a wrong move from a symmetric position, the algorithm should exploit this mistake and learn. As a result, symmetry should not be enforced if the opponent is not taking advantage of it.

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?

Greedy approach will not result in good agents. This is because the policy learnt by the agent consists of only a very small fraction of state space and a larger fraction of the state space is unexplored. There exists a trade-off between taking actions which enables us to get high rewards and taking arbitrary actions to explore all possible ways. This is referred to as the Explore-Exploit dilemma. And methods like $\epsilon - greedy$ search enables us to handle the situation in a better way.

Problems with greedy play is that the agent won't generalise well, when it is put in a random situation. This is because, the agent may not be aware of that particular state or what actions to take in that particular state, because it has never seen such a state before. This situation can be avoided by making sure that agent explores the state space.

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 set of probabilities. What 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?

With the step-size parameter reduced appropriately over time, but exploration rate is fixed constant, the probability obtained with "no learning from exploration" is the value of each state corresponding to the policy π followed i.e from each state the optimal action is taken, and it is the expected reward to be obtained in that case.

On the other hand, the probability obtained with "learning from exploration" is the value of each state obtained using a policy π' which is an active exploration policy. It accounts for greedy actions with exploration now and then. The value thus obtained is the expectation over all explored trajectories.

If we continue to make exploratory moves, the former is better to learn. This is because it is more consistent with the policy π followed. Also, updates due to exploration results in higher variance from sub-optimal states. This is because, due to exploration we might end up with a negative reward (due to a bad action) and hence the variance in that estimate is higher. As a result, the former results in lower variance in our estimates.

The "no learning from exploration" method results in more wins.

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-tactoe problem as posed?

Here, we are making updates to the value function after making moves against the opponent. Initial updates contribute less to learning the optimal value function. This is because, say if we follow $\epsilon - greedy$ approach, initial updates are due to exploration whereas later updates will ensure a larger fraction of state space is explored and hence are more meaningful. As a result, decaying/reducing the contribution of old updates can prove to improve performance/speed up learning.

Symmetries can be exploited if we are known that the opponent is taking advantage of it. This reduces time/memory for learning the RL agent and results in a better statistical solution.

Also, the exploration rate ϵ / step size α should be varied (decayed to 0 eventually). This ensures asymptotic convergence.