CS6700 ASSIGNMENT 1

Sabarinath N P *(CS10B020)*

Department of Computer Science and Engineering
IIT Madras

August 2013

Exercise

1.1 Question 1

Question: Suppose, instead of playing against a random opponent, the reinforcement learning algorithm described above played against itself. What do you think would happen in this case? Would it learn a different way of playing?

Answer: Yes, the agent will learn a different method of playing when compared to playing against a static model or nature. This is mainly due to fact that the opponent is also a learner and it changes its game models and policies in accord to our agent.

Also due to the concept of mixing exploration and exploitation in choosing the actions, one of the agent might perform better than the other. It all depends on the actions it chooses to take in different states.

Probability that both the agents play equally or both of them are learning at same rate is unlikely as there is some randomness in choosing the actions due to the process of exploration.

1.2 Question 2

Question: Many tic-tac-toe positions appear different but are really the same because of symmetries. How might we amend the reinforcement learning algorithm described above to take advantage of this? In what ways would this improve it? 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: With out loss of generality, it can be assumed that symmetric positions can be constructed by rotating the tic-tac-toe board in some direction. With this, the number of unique states in the tic-tac-toe reduces, as a result the size of the table that is used to store the values of each state reduces. Hence, the time taken by the greedy algorithm to exploit the moves already performed decreases. But representing two symmetric states with one number in the table might affect the agent because the path followed to achieve the two states might be different and hence the learning is might also be different.

If the opponent did not consider the symmetry, in that case our agent can either take advatage or not. It depends. One merit is the time taken by greedy algorithm decreases and the agent can learn about multiple states by one action due to the property of symmetry. The demerit is the agent will not learn by exploring the states whose symmetrically equivalent state is already explored. And again symmetrically equivalent positions need not have same value as the game might involve exploration in later stages. That is, the value of symmetric states can vary depending on the actions taken (exploitation or exploration) by the agent in the next state.

1.3 Question 3

Question: 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. Would it learn to play better, or worse, than a nongreedy player? What problems might occur?

Answer: It is safe to assume that the greedy agent never chooses to take exploratory actions. In that case, the greedy agent might have an edge when the two agents play their first few games. This is primarily due to the fact that the non-greedy player chooses to play exploratory action though it knows the best action for a given state. This introduces some randomness and also provides a way to learn about the different actions and their reward.

After some n ($n \gg 1$) games, the non-greedy agent would have learned about many actions and their rewards/values. But the greedy agent lacks this knowledge and it always chooses to play the best (local optima, not global optima) action. In this case the non-greedy agent has an edge as it can take the optimal (can be global optima) counter action for every action the greedy agent takes. So, over the course of time and number of games, non-greedy agent has greater chance of winning as it learned the complete knowledge of how the greedy agent plays and about the different actions to be taken.

1.4 Question 4

Question: Suppose learning updates occurred after all moves, including exploratory moves. If the step-size parameter is appropriately reduced over time, 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?

Answer: Exploratory moves are more often non-optimal (non-local optimal) actions for the given state as it is randomly choosen. The rewards resulted by those actions will be negative (or positive) though the value obtained after a series of actions might be positive (or negative). Hence, immediate learning from exploratory moves might give a wrong probability and feedback about the action taken. In this case, the learning from exploratory moves might result in less wins.

But when the exploratory moves result in the same sign of rewards and values, learning from these movess result in fast growth towards global optimum. Hence, it depends on how close the reward from exploratory moves are, to the actual 'values'.

1.5 Question 5

Question: 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: Few ways to improve the learning:

- To get feedback about all possible actions, the agent can act more biased towards the exploratory moves in the beginning. This might give a way to choose the best among the many possible actions instead of small set in the later stages. But this method might have bad performance in the beginning due to high randomness.
- When the number of states get larger, a pruning can be done similar to beam search to avoid storing the values of all possible states. This reduces the search time. Also, a data structure similar to priority queue can be maintained for fast access and update.

To improve the performance of the tic-tac-toe playing agent:

- Shallow depth search can be applied to look ahead few steps of the game considering the opposite player plays optimally. This handles the drawbacks of greedy algorithm by not choosing a bad local optima.
- After few games, few states can be marked a 'Loss' states, from which winning is impossible. This information gives a way to avoid making game losing exploratory moves. Similarly 'Win' states can be stored, to avoid not making game winning moves.
- After few games, in the later stages of a game making an irrelevant random exploratory action is not an good idea. So, introducing some bias towards choosing the random action so that it moves towards the goal (similar to A^* idea of moving towards goal) might decrease the number of losses.