DP&RL - Assignment 3

Chih-Chieh Lin (2700266) Iasmina Erdelean (2568500)

13 December 2020

Problem description

This assignment requires us to develop an MCTS tree search algorithm based on the game tic-tac-toe, so that we can reliably beat a random agent. This report would provides our solutions and results for solving the problem mentioned above. Regarding several questions mentioned in the assignment, the relevant results and solutions would be placed in the following sections.

1 Monte Carlo tree search

The MCTS algorithms need only a generative model of the environment (they are model-based):

$$x^{'}, r \sim \ G(x, a) \tag{1}$$

Some advantages include: the possibility of obtaining samples without having the whole transition function for the model in an explicit form; the optimal policy can be learned only where needed, so from current state x. The overall idea of MCTS methods is to estimate the action with the highest expected return:

$$V^*(x) = Q^*(x, a = \pi^*) = E^*_\pi[r_0 + \gamma r_1 + ...] \tag{2}$$

In our case, the rewards are only on the final states, thus winning results in a reward of 1 and loosing in a reward of -1.

In the initial phase of implementing the algorithm, we start with a root node and select a child node such that it picks the node with maximum win rate. We also want to make sure that each node is given a fair chance. The idea is to keep selecting optimal child nodes until we reach the leaf node of the tree. A good way to select such a child node is to use UCT (Upper Confidence Bound applied to trees) formula:

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}} \tag{3}$$

Then follows the expansion phase, so that when we reach a leaf node in the current tree Add one node with random rollout. After expansion, the algorithm picks a child node arbitrarily, and it simulates a randomized game from selected node until it reaches the resulting state of the game. Once the algorithm reaches the end of the game, it evaluates the state to figure out which player has won. It traverses upwards to the root and increments visit score for all visited nodes. It also updates win score for each node if the player for that position has won the playout. MCTS keeps repeating these four phases until some fixed number of iterations or some fixed amount of time.

2 Optimal Policy

In order to find the optimal policy, MCTS is a key algorithm by trying out trajectories from a state x in an MDP. This search is a combination of Monte-Carlo methods that rely on repeated random sampling in order to obtain numerical results and Tree search that relies on methods allowing the exploration of a tree data structure. Based on the theory, the MCTS converge to the optimal policy, because the number of trajectories tend to infinity. But our tic-tac-toe case can be solved with simpler

MCTS algorithms and at the same time have a reasonable number of computational steps. By reaching convergence, the optimal policy has been found.

3 Probability of Victory

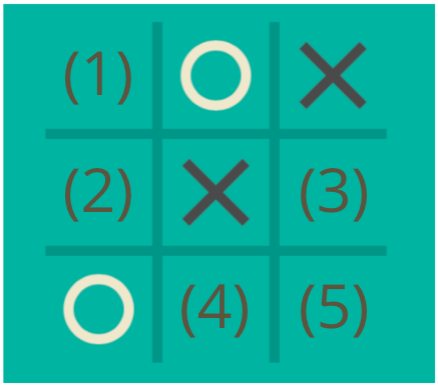

Figure 1: State in restricted condition

In the restricted condition shown Figure 1, we are asked to provide the probability of victory for all five possible actions. We notice that each possible action could generate 24 final game results. These results are able to derive by the tree structure. Afterwards, we find all the numbers of victory condition, and then divide it by all possible results (which is 24). Finally, the probability of victory is obtained and shown in Table1.

	(1)	(2)	(3)	(4)	(5)
MCTS: O action	0.417	0.583	0.333	0.417	0.583
MCTS: X action	0.5	0.583	0.667	0.333	0.75

Table 1: Probability of Victory calculated by tree concept

Furthermore, several simulation works are done to observe whether the action with higher probability of victory is a better choice for our MCTS agent. We first restrict the MCTS to take certain action(for example, place the circle on the top-left for the first action), and then continue the game until the game over. We do the simulation for 500 times, that is, the games started from the certain condition are played by MCTS and Random agent for 500 times. Afterwards, we can obtain the probability of victory by the simulation. The results are shown in Table2. It's not surprising that the probabilities we derived from simulation are different from the previous one because the MCTS would take the best actions for the rest of game. However, we still could observe that the trends or the distribution of the probability for both of them are similar.

	(1)	(2)	(3)	(4)	(5)
MCTS: O action	0.718	0.886	0.504	0.674	0.896
MCTS: X action	0.75	0.874	1	0.454	1

Table 2: Probability of Victory derived by simulation