# COMS W4701: Artificial Intelligence, Summer 2022

Homework 3 Solutions

## Problem 1: MCTS Practice (12 points)

(a) The resultant tree will have the same structure as the one provided. The updated win rates are as follows: root, 38/101; layer 2 node 1, 60/80; layer 3 node 2, 17/54; layer 4 node 1, 27/36; layer 5 node 1, 1/1. All other win rates are unchanged.

(b) The first child node will be selected, as it has the highest UCT value as computed below.

$$UCT(n_1) = \frac{60}{80} + \sqrt{\frac{\ln 101}{80}} = 0.99$$
$$UCT(n_2) = \frac{1}{10} + \sqrt{\frac{\ln 101}{10}} = 0.78$$
$$UCT(n_3) = \frac{2}{11} + \sqrt{\frac{\ln 101}{11}} = 0.83$$

(c) Node $n_1$ will continue being selected for a number of iterations, and its UCT value will decrease each time as the denominators increase. The third child node will be selected once $UCT(n_1)$ becomes smaller than $UCT(n_3)$. This occurs when the following equality holds:

$$\frac{60}{80+N} + \sqrt{\frac{\ln(101+N)}{80+N}} = \frac{2}{11} + \sqrt{\frac{\ln(101+N)}{11}}$$

The solution is $N = 17$ iterations later.

(d) In this case, we want to solve for the value of $\alpha$ that would make the UCT values of $n_1$ and $n_3$ equal using original the win rates:

$$\frac{60}{80} + \alpha\sqrt{\frac{\ln 101}{80}} = \frac{2}{11} + \alpha\sqrt{\frac{\ln 101}{11}}$$

The solution is $\alpha = 1.394$. It is higher than the original value because this encourages exploration of a suboptimal action as opposed to exploitation of the current best action.

## Problem 2: Mini-Blackjack (12 points)

(a) You should have a DAG with seven nodes. 0 has edges to 2, 3, 4; 2 has edges to 4, 5, 6; 3 has edges to 5, 6, done. All transitions have probability $\frac{1}{3}$. Node 4 points to 6 and done with probabilities $\frac{1}{3}$ and $\frac{2}{3}$; both 5 and 6 only point to done with probability 1 each.

(b) In states 5 and 6, optimal actions are obviously to stop, giving rewards 5 and 6. Then we can solve for each of the remaining states by comparing both actions:

- $V^*(4) = \max(4, \frac{0.9}{3}(6)) = 4$, $\pi^*(4) = \text{stop}$
- $V^*(3) = \max(3, \frac{0.9}{3}(5+6)) = 3.3$, $\pi^*(3) = \text{draw}$
- $V^*(2) = \max(2, \frac{0.9}{3}(4+5+6)) = 4.5$, $\pi^*(2) = \text{draw}$
- $V^*(0) = \max(0, \frac{0.9}{3}(4.5+3.3+4)) = 3.54$, $\pi^*(0) = \text{draw}$

Dynamic programming is not needed here because there no set of state values depends on each other in both directions; if $s'$ is a descendent of $s$, we can be sure that $s'$ is not an ancestor of $s$ along another path. In other words, the state transition diagram is a DAG.

(c) We want to solve for when $Q(3, \text{stop}) = Q(3, \text{draw})$:

$$3 = \frac{\gamma}{3}(5+6)$$

which gives us $\gamma = \frac{9}{11}$. However, no nonzero value of $\gamma$ would change the optimal action at state 0, since it is always possible to get some nonzero positive utility by drawing vs stopping.

# Problem 3: Dynamic Programming (12 points)

(a) $V_1(s) = s$, since the stop action always returns the current card sum and that is always greater than or equal to the initial values $V_0(s) = 0$.

(b) Since we have the values, we simply need to do policy improvement. Given the current values, it is always better to **stop**. So $\pi_1(s) =$ **stop** for all $s$ except for $s = 0$. For the latter, both the draw and stop actions give utility 0, giving us a tie, so we have $\pi(0) =$ **draw**.

(c) The results are as follows:

- $V_2(6) = \max(6, 0) = 6$
- $V_2(5) = \max(5, 0) = 5$
- $V_2(4) = \max(4, \frac{0.9}{3}(6)) = 4$
- $V_2(3) = \max(3, \frac{0.9}{3}(5+6)) = 3.3$
- $V_2(2) = \max(2, \frac{0.9}{3}(4+5+6)) = 4.5$
- $V_2(0) = \max(0, \frac{0.9}{3}(2+3+4)) = 2.7$

The values (specifically $V(0)$) have not converged.

(d) Again, we already have the values for $\pi_1$, so we simply perform policy improvement. This will follow the comparisons of $V_2$ above, but using argmax rather than max. We thus obtain $\pi_2(s) =$ stop for $s = 4, 5, 6$ and $\pi_2(s) =$ draw for $s = 3, 2, 0$. This is indeed the optimal policy.

# Problem 4: Reinforcement Learning (12 points)

(a) MC updates state values based on returns obtained at the end of an episode. The updates are as follows:

- $V(0) = \frac{1}{5}(0+0+6+5+5) = 3.2$
- $V(2) = \frac{1}{2}(0+5) = 2.5$
- $V(3) = \frac{1}{2}(0+5) = 2.5$
- $V(4) = \frac{1}{2}(0+6) = 3$
- $V(5) = 5$
- $V(6) = 6$

The order of the episodes does not matter for MC, as we are simply averaging the returns from each episode separately.

(b) TD will update values incrementally once we start seeing rewards. This starts at the end of episode 3, and the updates are as follows:

- $V(6) \leftarrow 0 + 0.8(6-0) = 4.8$
- $V(5) \leftarrow 0 + 0.8(5-0) = 4$

- $V(2) \leftarrow 0 + 0.8(4 - 0) = 3.2$
- $V(5) \leftarrow 4 + 0.8(5 - 4) = 4.8$

All other state values, as estimated by TD, are 0. The episode ordering does matter for TD, as we are using any successor state values estimated from previous transitions to update values in later transitions.

(c) The Q values that are updated are $Q(6, stop)$, $Q(5, stop)$, and $Q(2, draw)$. The exploratory actions are the two stop actions in episodes 3 and 4.

# Problem 5: "Real" Blackjack

## 5.3 DP Analysis (9 points)

(a) The rightmost linear segment is a result of the optimal action being stop at the higher states. The value of each state is simply equal to the card sum. The middle curve reflects the values of the states from which it is possible to draw once before stopping, The values are thus the expected value of the successor state. Lastly, the leftmost curve is discontinuous with the middle one because from the two card sums 0 and 1 it is possible to draw a card **twice**. The result is a much higher expected value than simply drawing once.

(b) As $\gamma$ decreases, the values of the states whose optimal action is to draw decrease as well, since future rewards are worth less. In the "border" states like 10 or 11 where it was previously optimal to draw, we may switch to the stop action instead; as $\gamma$ decreases, more and more states switch to the stop action to collect the reward *now* rather than wait for a reward *later*. The value function now appears to consist of two segments; the original first two segments merge into one (corresponding to states with the draw action), since a second draw action starting from the 0 or 1 state is worth so little compared to the first one.

(c) For negative living rewards, we see that the first two segments of the value function (originally corresponding to draw) start to shift downward. This is because the state values now incorporate the negative rewards. For even more negative values of $lr$, the optimal action becomes stop in more and more states, becoming a universal policy around $lr = -7$. The opposite happens when the $lr$ is positive. All segments of the value function increase at different rates, because each additional draw action contributes more positive $lr$. At $lr > 21$, it becomes advantageous to draw at every state, since the $lr$ is guaranteed is greater than the maximum possible stop action reward.

## 5.5 RL Analysis (9 points)

(a) With $\varepsilon = 0$, we never explore. Since all Q values are initially 0, either action can be considered to be the exploit action If you globally used stop, then we would never visit any other state. If you globally used draw, then we end drawing in every state and eventually reach every other state, since Q values can only increase. In general, larger $N$ means more visits to each state. In particular, the 0 state is visited the most as the starting state, followed by 10 and then by 20 due to greater chances of drawing a 10 value card. The number of visits otherwise increases gradually to and decreases gradually from 10.

(b) Exploration ensures that we eventually try all states and learn their optimal actions. Again assuming default tiebreaker of stop, $\varepsilon = 0$ means we never leave the 0 state and receive any reward. Otherwise, a larger value of $\varepsilon$ will lead to a slower increase in cumulative reward. All reward curves increases approximately linearly after we have approximately learned the optimal policies. However, the rate at which we receive rewards is lowered by a higher chance for exploration.

(c) A learning rate that is too low may significantly lower the convergence rate. In our plots we see that the value of 21 is much lower than what it should be (21) for $\alpha = 0.1$. This is because it is the furthest value from the initial guess of 0, requiring many updates before it gets close to 21. A learning rate that is too high makes the incremental updates too large, giving a lot of variance to the estimated values. We see rather large variations in the Q values for large $\alpha$ as compared to low $\alpha$ which are more stable.