CS6700 - Reinforcement Learning - Assignment 1

Atul Balaji

February 15, 2019

1 Question 1

The tic-tac-toe game, due to the board layout has four axes of symmetry. Therefore, it is possible to achieve a much more **compact state representation** by bringing it down to a quarter of the size, by taking advantage of the symmetry. This would result in a significant **reduction of memory and increase in speed** of the algorithm.
If the opponent is also taking advantage of symmetry, we must do so too. However, if the **opponent does not take advantage of symmetry, we should not**. If we use symmetry, we will be enforcing a symmetry rule which the opponent may not obey. For example, the opponent may always make a wrong move at some particular corner and play correctly elsewhere. If we use a symmetric representation, we lose the ability to differentiate among what we see as symmetric positions. We will not be able to exploit this information and learn, resulting in lower chances of winning and a decrease in performance.

2 Question 2

When an agent plays the game against itself, we can view it as a game between two similar agents 1 and 2. For a given board position, the state representation seen by the two agents is different. Also, a WIN for agent 1 is LOSE for agent 2 and vice-versa. If we define the rewards to be +1 for WIN, 0 for DRAW and -1 for LOSE, both agents will learn to draw the game in most cases. Given a board position where agent 1 can either WIN or DRAW, agent 2 can LOSE or DRAW. Then, agent 2 will learn the optimal set of actions to draw the game. As more moves are made, both agents continue to adapt until an equilibrium is reached, which may be fixed (same state) or cyclic (both agents repeating same set of moves). The agent learns from its moves and becomes incrementally better with more updates. It reaches a higher skill level when playing against itself.

3 Question 3

Greedy play will result in the agent playing worse. In the starting stages, the policy learned by the agent includes only a small fraction of the states, leaving most of it unexplored. If the agent always takes a greedy action (the one with maximum estimated reward), it may be stuck in some state which it believes to be optimal, even though there may exist better states which have not been explored, especially if there is a large number of possible actions. Instead of always playing greedy, a non-greedy player that achieves a good balance in the **Explore-Exploit trade-off** will perform better.
The main problem that occurs when an agent plays in a greedy manner is the inability to generalize. When put in a new state which it has not explored while learning, the agent will not know what action to take, since it has not visited that state before. This results in poorer performance.

4 Question 4

In this question, the initial values for all arms have been set to +5 instead of 0, which is a very optimistic initialization. To start off, the agent will think that all the arms are optimal due to the initialization and will have to sample each arm multiple times until it finds that some of them do not have a good reward. This will have to continue until the values of the bad arms drop below that of the optimal arm. This is the reason for the oscillations and spikes in the initial part of the curve.
This method may perform better on some early steps if the values of the bad arms are quite far from that of the best arm and they soon drop below the best arm. A lower variance in the value distribution will also result in better performance by the greedy algorithm, since it can find the best arm with more ease. It performs worse if the true values of a significant number of arms are close to each other and also if the variances in the values of arms are also high.

5 Question 5

In this question, the rewards for different arms are sampled from some unknown distribution. But, after the agent picks an arm, we get to know the **rewards of all the arms** in that time step.
In this scenario, **exploration is not necessary**, since rewards of all arms at a particular time are known to us once we pull any arm. Existing regret minimization algorithms such as UCB will not be optimal since it does not take advantage of the information about the rewards of all other arms in a single pull of an arm. Also, ϵ-greedy algorithm is also not optimal since we need not explore here.
The best algorithm for this case would be the **greedy approach** where at every step, we select the arm with the best estimated reward till that point.

6 Question 6

We know the expected payoffs of all the arms beforehand. But we don't know the mapping between each arm to its expected payoff. Expected payoffs are - 4.6, 3.1, 2.3, 1.2, 0.9 for the given problem.
Let $\Delta_i = q_*(a_*) - q_*(a_i)$. An improved version of UCB can be used, where we choose the arm that maximizes $Q_j + \sqrt{\frac{2lnt}{n_j}}\Delta_j$. This achieves better bounds than UCB.
In order to minimize regret, we must find the best action in the smallest number of time steps and then onwards, select the same action in a greedy manner. If $u_j = \sqrt{\frac{2lnt}{n_j}}$ is below $0.75 = (4.6 - 3.1)/2$, then it means our uncertainty is low enough, so that the best action can be distinguished from the next best. So we can conclude that the arm with the highest estimate is in fact the optimal arm. We need to sample until u_j goes below 0.75, and then on take greedy action.

7 Question 7

Given: n arms, $\pi_t(a)$, $\rho_t(a)$
Using the REINFORCE procedure, we update the parameters as:

$$\Delta\rho_t(a_t) = \beta(r_t - \overline{r_t})\frac{\partial \ln \pi(a_t;\rho)}{\partial \rho_t(a)}, \tag{1}$$

where a_t is the action taken at time t.
The characteristic eligibility term (partial derivative) can be written as:

$$\frac{\partial \ln \pi(a_t;\rho)}{\partial \rho_t(a)} = \frac{\partial}{\partial \rho_t(a)} \ln \frac{e^{\rho_t(a_t)}}{\sum_{b=1}^{n} e^{\rho_t(b)}} = \frac{\partial}{\partial \rho_t(a)}(\rho_t(a_t) - \ln(\sum_{b=1}^{n} e^{\rho_t(b)})) = 1 - \pi_t(a_t;\rho) \tag{2}$$

Now, r_t is the reward received at time t and the baseline $\overline{r_t}$ is the average of all rewards before t.
The update conditions is:

$$\rho_{t+1}(a_t) - \rho_t(a_t) = \Delta\rho_t(a_t) = \beta(r_t - \overline{r_t})(1 - \pi_t(a_t;\rho)) \tag{3}$$

8 Question 8

Given:
Parameters are μ and $\sigma^2 = v$ of the normal distribution according to which actions are sampled, baseline = 0.

$$\pi(a;\mu,v) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(a-\mu)^2}{2v}} \tag{4}$$

For μ:

$$\frac{\partial \ln \pi(a_t;\mu_t,v_t)}{\partial \mu_t} = \frac{\partial}{\partial \mu_t}\left(-\frac{(a_t-\mu_t)^2}{2v_t}\right) = \frac{a_t - \mu_t}{v_t} \tag{5}$$

For v:

$$\frac{\partial \ln \pi(a_t;\mu_t,v_t)}{\partial v_t} = \frac{\partial}{\partial v_t}\left(-\ln(\sqrt{2\pi v_t})\right) + \frac{\partial}{\partial v_t}\left(-\frac{(a_t-\mu_t)^2}{2v_t}\right) \tag{6}$$

Solving,

$$\frac{\partial \ln \pi(a_t;\mu_t,v_t)}{\partial v_t} = -\frac{1}{2v_t} + \frac{(a_t-\mu_t)^2}{2v_t^2} = \frac{1}{2v_t}\left[\frac{(a_t-\mu_t)^2}{v_t} - 1\right] \tag{7}$$

Thus, the updates are:

$$\mu_{t+1} = \mu_t + \beta r_t \left(\frac{a_t - \mu_t}{v_t} \right) \tag{8}$$

$$v_{t+1} = v_t + \beta r_t \frac{1}{2v_t} \left[\frac{(a_t - \mu_t)^2}{v_t} - 1 \right] \tag{9}$$