CS6700 Reinforcement Learning
Written Assignment #1

Sai Vinay G
CE17B019
Indian Institute of Technology Madras

February 17, 2019

Problem 1

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?

- Using symmetry we can remove all the similar positions and store only one equivalently similar position.

- During learning process we can reduce the memory usage to store positions and the time required by the agent to search for next move, as the agent now has lesser number of positions which can describe all the states possible.

- If opponent does not take advantage of symmetries i.e, when there are two similar states s1, s2 the opponent takes moves, which return two dissimilar states say s3 and s4. Suppose there were good and bad moves possible in s1 and s2 for the opponent to make. If the opponent makes a good move in s1 and a bad move while in s2(as our opponent does not take advantage of symmetry and considering opponent is not a good player). Thus, the agent can exploit the above bad move to increase the reward. So, its better to keep separate values for symmetrical states.

- No, as described above there is a possibility to get different values for symmetrical states depending on our opponent.

Problem 2

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?

- Here the main difference is that the agent is learning from both sides i.e, Both initially make random moves which results in winning from one side and losing from the other or draw is also possible. Each side is trying to make a better policy against an opponent, who is in turn trying to make a better policy against it. So, both will learn gradually to make better moves as both agents (itself) encounters wins, loses, draws and they keep updating their policies to become better after each update. Eventually, the policies may converge to optimal policy.
- Which will not be same as the policy learned when played against random opponent who is not learning while playing as, the agent will learn to make a policy which converges to optimal policy against the opponent.

Problem 3

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 non-greedy player? What problems might occur?

- It will not be able to play better than a non-greedy player as it does explore much and only has the policy for small set of states.
- The greedy hasn't seen lot of states. So, it might not be able to learn the optimal moves possible from a state and when a new state appears the greedy agent doesn't know what to do as it did not see the state before. These can be tackled by making the agent to explore more.

- As, the greedy agent doesn't explore more, it may get stuck with sub optimal moves and keep on playing those, where there are better optimal moves possible.

Problem 4

The results shown in Figure 2.3 (of course text book uploaded in moodle) should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps?

- Initially as each arm was having high expected value (+5), optimal method chooses each of the arm once and moves to next as the expected reward after once pulling decreases as the maximutm possible for each arm is is very less(as they are sampled from mean=0 and variance=1). But, as returns from the optimal arm more likely to be higher than other arms, So, 40% of the times optimal arm was chosen (reason for spikes). But, after pulling the optimal arm for few number of times the expected reward still decreases. As, the other arms even though being not optimal, all were initialized to give an expected reward of +5 and they were pulled less number of times(as compared to optimal)so their expected rewards are be still higher than the optimal arm which is pulled for some number of times. So, this switching between optimal and not optimal action is cause of oscillations.

- This method better because it encourages enough exploration and then after finding optimal arm it doesn't explore, which is the cause for better performance as, ϵ - greedy even after finding the optimal arm, it still explore which results in decrease of performance.

- As, it explores rapidly in the early steps, it is more likely that it finds the optimal arms whereas, the ϵ - greedy explores much slowly. So, the optimistic approach has better early performance.

Problem 5

Define a bandit set up as follows. At each time instant for each arm of the bandit we sample a reward from some unknown distribution. Now the agent picks an arm. The environment then reveals all the rewards that were chosen. Regret is now defined as the difference between the best arm at that instant and the one chosen summed over all times steps. Would the existing algorithms for bandit problems work well in this setting? Can we do better by taking advantage of the fact that all rewards are revealed? For e.g., exploration is not an issue now, since all arms are revealed at each time step.

- The existing algorithms will work well in this case. Here, we can take the advantage of the fact that environment is revealing the rewards that would have got by the agent only if it had chosen that arm.
- Previously, in order to find the reward that an arm gives we had to pull it and then check. In this case its easy to find the optimal arm quickly as in each move we will be knowing the rewards obtained from all the arms.(i.e, previously to get 5 reward values from each arm we had to pull a total of 5*(no of arms) times, but now it can be done only by pulling 5 times).
- So, now we keep track of the rewards of all arms, every time we pull any arm and find the expected reward of each arm at every step and chose the arm which has maximum expected reward to pull for next time.

Problem 6

Consider a bandit problem in which you know the set of expected payoffs for pulling various arms, but you do not know which arm maps to which expected payoff. For example, consider a 5 arm bandit problem and you know that the arms 1 through 5have payoffs 3.1, 2.3, 4.6, 1.2, 0.9, but not necessarily in that order. Can you design a regret minimizing algorithm that will achieve better bounds than UCB? What makes you believe that it is possible? What parts of the analysis of UCB will you modify to achieve

better bounds? Note that I am not asking you for a complete algorithm or analysis, only the intuition.

- As the difference in the expected payoffs is high. So, it is possible to design a regret minimizing algorithm achieving better bounds than UCB.
- In general UCB we first try all arms once and then loop over such that, at each step we select an arm j which maximizes $\left(Q(j) + \sqrt{\frac{2lnn}{n_j}}\right)$, where the first term is the estimate of the arm, the second term in the expression is the uncertainty in the estimate.
- If for an arm j we have $\left(Q(j) - \sqrt{\frac{2lnn}{n_j}}\right) > 3.1$, implies that with high confidence we are sure that our expected value's lower bound is greater than 3.1. Hence, as there is only one arm which has expected payoff greater than 3.1 i.e, 4.6, which is the optimal arm and in between, we can keep on elimination the sub optimal arms i.e, if for arm j we have $\left(Q(j) + \sqrt{\frac{2lnn}{n_j}}\right) < 4.6$, and we will be left with optimal arm quickly.
- After we get the optimal arm we discard the other arms and select only the optimal which reduces the regret as we found out the optimal quicker than normal UCB.

Problem 7

Consider a bandit problem in which the parameters on which the policy depends are the preferences of the actions and the action selection probabilities are determined by the softmax relationship as $\pi_t(a) = \frac{e^{\rho_t t}}{\sum_{b=1}^{n} e^{\rho_t(b)}}$, where n is the total number of actions and $\rho_t(a)$ is the preference value of action a at time t. Derive the parameter update conditions according to the REINFORCE procedure considering the above described parameters and where the baseline is the reference reward defined as $\overline{r_{t+1}} = \overline{r_t} + \beta[r_t - \overline{r_t}]$, where r_t is the reward received at time t and β is the step size parameter.

According to REINFORCE procedure we update parameters as,

$$\theta_{t+1} \leftarrow \theta_t + \Delta\theta_t$$
$$where\ \Delta\theta_t = \alpha_t(r_t - b_t)\nabla_\theta ln\ \pi(a_t;\theta)$$

Here, the parameters are the preferences of the actions. So, we get

$$\Delta\rho_t(a) = \alpha_t(r_t - b_t)\nabla_{\rho_t(a)} ln\ \pi(a_t;\rho_t(a)) \quad a\ is\ arbitrary\ action$$
$$Given\ reinforcement\ baseline\ is\ \overline{r_{t+1}} = \overline{r_t} + [r_t - \overline{r_t}]$$
$$Given\ \pi(a_t) = \frac{e^{\rho_t(a_t)}}{\sum_{b=1}^{k} e^{\rho_t(b)}}$$

Lets first evaluate $\nabla_{\rho_t(a)} ln\pi(a_t;\rho_t(a))$

$$\begin{aligned}
ln\pi(a_t;\theta) &= ln(e^{\rho_t(a_t)}) - ln(\sum_{b=1}^{k} e^{\rho_t(b)}) \\
&= \rho_t(a_t) - ln(\sum_{b=1}^{k} e^{\rho_t(b)}) \\
\nabla_{\rho_t(a)} ln\pi(a_t;\theta) &= \frac{\partial ln\pi(a_t;\theta)}{\partial \rho_t(a)} \\
&= \mathbb{1}_{a_t=a} - \frac{e^{\rho_t(a)}}{\sum_{b=1}^{k} e^{\rho_t(b)}} \\
&= \mathbb{1}_{a_t=a} - \pi_t(a) \\
\implies \Delta\rho_t(a) &= \alpha_t(r_t - b_t)(\mathbb{1}_{a_t=a} - \pi_t(a)) \\
Given\ b_t = \overline{r_{t-1}} + \beta[r_{t-1} - \overline{r_{t-1}}] & \\
\therefore \Delta\rho_t(a) &= \alpha_t(r_t - \overline{r_{t-1}} - \beta[r_{t-1} - \overline{r_{t-1}}])(\mathbb{1}_{a_t=a} - \pi_t(a))
\end{aligned}$$

The update equation is :

$$\rho_{t+1}(a) \leftarrow \rho_t(a) + \alpha_t(r_t - \overline{r_{t-1}} - \beta[r_{t-1} - \overline{r_{t-1}}])(\mathbb{1}_{a_t=a} - \pi_t(a))$$

Problem 8

Repeat the above problem for the case where the parameters are the mean and variance of the Normal distribution according to which the actions are selected and the baseline is zero.

- Given the parameters are mean μ_t and variance σ^2 of a normal distribution.
- The action taken is a continuous random variable with normal distribution.

According to REINFORCE procedure we update parameters as,

$$\theta_{t+1} \leftarrow \theta_t + \Delta\theta_t$$
$$where\ \Delta\theta_t = \alpha_t(r_t - b_t)\frac{\nabla_\theta \pi(a_t;\theta)}{\pi(a_t;\theta)}$$

Here probability of selecting an action a_t is,

$$\pi(a_t) = \int_{a_t} \frac{1}{\sqrt{2\pi\sigma_t^2}} e^{-\left(\frac{a_t - \mu_{tt}}{\sqrt{2}\sigma_t}\right)^2} da$$

- Lets first derive the update rule for μ_t

$$\Delta\mu_t = \alpha_t(r_t - b_t)\frac{\nabla_{\mu_t} \pi(a_t;\mu_t)}{\pi(a_t;\mu_t)}$$

Lets evaluate $\frac{\nabla_{\mu_t} \pi(a_t;\mu_t)}{\pi(a_t;\mu_t)}$

$$
\begin{aligned}
\frac{\nabla_{\mu_t}\pi(a_t;\mu_t)}{\pi(a_t;\mu_t)} &= \frac{\nabla_{\mu_t}\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= \frac{\int_{a_t}\nabla_{\mu_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_{tt}}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= \frac{\int_{a_t}\frac{\partial}{\partial\mu_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_{tt}}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= \left(\frac{a_t-\mu_t}{\sigma_t^2}\right)\frac{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_{tt}}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= \left(\frac{a_t-\mu_t}{\sigma_t^2}\right)
\end{aligned}
$$

Given baseline = 0

The update equation for μ is :

$$\mu_{t+1} \leftarrow \mu_t + \alpha_t r_t\left(\frac{a_t-\mu_t}{\sigma_t^2}\right)$$

- Now the update rule for σ_t

$$\Delta\sigma_t = \alpha_t(r_t - b_t)\frac{\nabla_{\sigma_t}\pi(a_t;\sigma_t)}{\pi(a_t;\sigma_t)}$$

Lets evaluate $\frac{\nabla_{\sigma_t}\pi(a_t;\sigma_t)}{\pi(a_t;\sigma_t)}$

$$\begin{aligned}
\frac{\nabla_{\sigma_t}\pi(a_t;\sigma_t)}{\pi(a_t;\sigma_t)} &= \frac{\nabla_{\sigma_t}\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= \frac{\int_{a_t}\nabla_{\sigma_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_{tt}}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= \frac{\int_{a_t}\frac{\partial}{\partial\sigma_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_{tt}}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= -\frac{1}{\sigma_t}\left[1+\frac{(a_t-\mu_t)^2}{\sigma_t^2}\right]\frac{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_t}{\sqrt{2}\sigma_t}\right)^2}da}{\int_{a_t}\frac{1}{\sqrt{2\pi\sigma_t^2}}e^{-\left(\frac{a_t-\mu_{tt}}{\sqrt{2}\sigma_t}\right)^2}da} \\
&= -\frac{1}{\sigma_t}\left[1+\frac{(a_t-\mu_t)^2}{\sigma_t^2}\right]
\end{aligned}$$

The update equation for σ is :

$$\sigma_{t+1} \leftarrow \sigma_t - \alpha_t r_t \frac{1}{\sigma_t}\left[1+\frac{(a_t-\mu_t)^2}{\sigma_t^2}\right]$$