CS533
Intelligent Agents and Decision Making
Written Assignment #3

1. Consider the algorithm described on Slide 18 (titled "Revision to Naive Approach") of the model-based RL slides. Give an example of a specific MDP that can lead the algorithm to converge to a sub-optimal policy. Justify your answer by describing a specific execution of the learning algorithm where the sub-optimal convergence occurs. The initial model (step 1 of the algorithm) should set all rewards to zero and if a state s has never been encountered, then the transition distributions for all actions in s should be initialized to uniform. It should be possible to construct a very small MDP for this problem.

   Solution: The following slides 19 and 20 have an example, where the algorithm converges to a sub-optimal policy. Make sure that this example is understood. It should also be possible to create an even simpler example.

2. [**Based on Exercise 6.12 from Sutton & Barto**] Consider the Q-learning and SARSA learning algorithms and suppose that action selection used during learning is greedy—i.e. the action selected at each step is the one that maximizes the Q-function estimate. Is Q-learning then exactly the same algorithm as Sarsa? Will the algorithms make exactly the same action selections and weight updates? Is there anything wrong with this way of learning?

   Solution: Recall the SARSA and Q-learning updates:

   $$\begin{aligned} Q(s,a) &\leftarrow Q(s,a) + \alpha\left(r + (\beta Q(s',a') - Q(s,a)\right) \text{ SARSA} \\ Q(s,a) &\leftarrow Q(s,a) + \alpha\left(r + \beta \max_{a' \in A} Q(s',a') - Q(s,a)\right) \text{ Q-learning} \end{aligned}$$

   Here r and s' are the reward and state that are observed after taking action a in state s. Also, for SARSA, a' is the action taken by the "behavior policy" (i.e. the policy followed during learning). Note that if the behavior policy is greedy then $a' = \arg\max_{b \in A} Q(s',b)$. Plugging this into the SARSA update gives us exactly the Q-learning update.

   Of course this would be a very bad learning algorithm, for the same reasons as the previous question. If the algorithm does not explore during learning and instead always acts greedily, then it can get locked into a sub-optimal policy for which the gathered experience does not support policy improvement.

3. Suppose that you run the SARSA learning algorithm, but instead of using a GLIE explore/exploit policy, you use a completely random exploration policy. That is, SARSA is run when learning from episodes generated by uniformly random action selection. What will the Q-function learned by SARSA converge to in this situation? You can assume that the learning rate is decayed at a standard theoretically correct rate.

   Solution: Recall that the SARSA algorithm is fundamentally a policy evaluation algorithm. That is, if it follows a policy π then it will converge to Q_π, i.e. the true Q-function of π. In this case, SARSA is following a fixed policy π_r that randomly selects actions. Thus, SARSA will converge to the Q-function of the random policy where $Q_{\pi_r}(s,a)$ tells us the value of selecting a in state s and then acting randomly thereafter.

4. Repeat the previous problem, but instead of SARSA consider the Q-learning algorithm.

   Solution: As we discussed in class and as described in the book, Q-learning will converge to the optimal Q-function $Q^*(s,a)$ regardless of the behavior policy during learning, as long as

all states-action pairs are tried infinitely often. As long as the MDP we are in does not have dead-ends that prevent the random policy from exploring all state-action pairs, the random policy satisfies the requirements of Q-learning for convergence to the optimal policy. Note that this property of Q-learning is due to the fact that it can be derived from asynchronous Value Iteration, which also does not care about the order that states are updated.

5. Why might one prefer to use a Boltzmann exploration policy rather than an ϵ-greedy exploration policy?

   Solution: ϵ-greedy exploration only distinguishes between the current greedy action and other actions. That is, when ϵ-greedy takes and exploration step it selects uniformly among the actions with no preference of one over another. However, it makes sense to eventually have a preference for exploring a action that appears slightly sub-optimal more than an action that appears very sub-optimal. The Boltzmann exploration policy does this by assigning probabilities to actions that are ordered according to their current Q-value estimates. So if we have two actions a and b and the Q-value estimate is higher for a than b, the Boltzmann policy will select a with a higher probability than b. The temperature controls how much a difference in the Q-values translates to a difference in the probabilities. A larger temperature will tend to result more uniform (i.e. closer) probabilities.

6. Consider an MDP where all policies are guaranteed to enter some terminal state, which ends an episode. Assume that terminal states either have a reward of 1 or reward of 0 and that all other rewards in the MDP are zero. Suppose that we have a policy π and wish to use Monte Carlo to estimate $V_\pi(s_0)$ for an initial state with no discounting ($\beta = 1$). Note that the value of $V_\pi(s_0)$ is equal to the probability that π reaches a terminal state with reward 1 when started in s_0 (make sure that this makes sense). How many episodes of the policy starting from s_0 must we observe in order to guarantee that with probability at least 0.99 the estimate $\hat{V}(s_0)$ of $V_\pi(s_0)$ is within 0.01 of the true value? You will want to use the Chernoff bound for this purpose.

   Solution: If we consult the Chernoff bound slides (slides 11-14 in RL for Policy Evaluation Slide Deck), we might consider using the final result on slide 14, but our setup here uses $\beta = 1$, which is valid because we have assumed that all policies will eventually enter a terminal state, so all values are bounded by 1. The result on slide 14 was for the general infinite horizon setting and is only meaningful for $\beta < 1$. So for this problem we need to start with the original statement of the Chernoff bound.

   We can view each episode of the policy in this MDP as flip of a coin with probability of heads $p_h = V_\pi(s_0)$, where a flip of heads corresponds to terminating with a reward of 1 and a flip of tails means terminating with a reward of 0. Our Monte-Carlo estimate $\hat{p}_h = \hat{V}(s_0)$ is just the average of w episodes/flips. Note that this coin flip scenario is exactly what was analyzed on slide 12 of the RL for Policy Evaluation slide deck. There we saw that if we flip a coin w times (run w episodes) then the event $|\hat{p}_h - p_h| \geq \epsilon$ has probability of occurring less than $\exp\left(-\epsilon^2 w\right)$. In our problem we would like to ensure that $|\hat{p}_h - p_h| < 0.01$ with probability at least 0.99. This is equivalent to ensuring that $|\hat{p}_h - p_h| \geq 0.01$ occurs with probability no greater than $1 - 0.99 = 0.01$. This matches the form of the Chernoff bound with $\epsilon = 0.01$. We can now find the w that provides our guarantee by making sure the right side of the Chernoff bound is less than 0.01. That is,

$$\exp\left(-\epsilon^2 w\right) \leq \delta$$

with $\epsilon = 0.01$ and $\delta = 0.01$. Solving for w we get:

$$w \geq \frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)$$

where the log is the natural logarithm. Plugging in the values of ϵ and δ gives: $w \geq 46051.70$. So to get a true guarantee we would need around $46K$ samples. That seems like a lot, but that is typical when using these types of bounds that are guaranteed to hold for any distribution of data. We can see from the bound on w that ϵ is the dominant factor. If we change are goal to guaranteeing that we are within 0.1 of the true value with probability at least 0.99, the sample requirement drops by 2 orders of magnitude $w > 460.5$, which is beginning to seem more reasonable.