CS6700 : Reinforcement Learning
Written Assignment #1

Intro to RL, Bandits, DP — Deadline: 23 Feb 2020, 11:55 pm

Name: Prajjwal Kumar — **Roll number:** NA16B115

- This is an individual assignment. Collaborations and discussions are strictly prohibited.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- Type your solutions in the provided LaTeXtemplate file.
- **Please start early.**

1. (2 marks) You have come across Median Elimination as an algorithm to get $(\epsilon, \delta)-$PAC bounds on the best arm in a bandit problem. At every round, half of the arms are removed by removing arms with return estimates below the median of all estimates. How would this work if we removed only one-fourth of the worst estimated arms instead? Attempt a derivation of the new sample complexity.

Solution:
In this question, we are showing a median elimination algorithm with removal of only one-fourth of the worst estimated arms in each round leading to one optimal arm at last instead of the half.

We know that it's a $(\epsilon, \delta)-$PAC algorithm i.e. it gives an ϵ-optimal arm with (1 - δ) probability.

So, for working of this algorithm, we need to find
1. No of times to sample every arm in a time step.
2. ϵ update
3. δ update
4. Sample complexity

1.1 Notation:
n = No. of arms
l = Round no.
s = No of times to sample every arm in a time step
a^* = Optimal arm
p_i = Estimate value of non-optimal arm i
p_i^* = True value of non-optimal arm i

$p(a^*)$ = Estimate value of optimal arm
$p^*(a^*)$ = True value of optimal arm

1.2 No. of Samples:
Since, it's a (ϵ, δ)-PAC algorithm, we can consider no of samples to be:

$$s = \frac{\ln(\frac{y}{\delta_l})}{(\frac{\epsilon_l^2}{2})}$$

Now, we will find failure probability i.e. probability of eliminating the optimal arms in earlier rounds. It must be upper bounded by δ. So, there are two events of failure.

A.When estimate value of optimal arm is under-estimated.

$$E_1 = [p(a^*) < p^*(a^*) - \frac{\epsilon_l}{2}]$$

Using **Chernoff-Hoeffding bound** and sample no. s definiton,

$$P[E_1] <= e^{-2*\epsilon_l^2*s} <= \frac{\delta_l}{y}$$

B.When estimate value of some non-optimal arms is over-estimated.

This is considered when E_1 doesn't hold. So, We find the probability of estimated value of a non-optimal arm more than that of the optimal arm.

$$P[(p_i > p(a^*)|p(a^*) >= p^*(a^*) - \frac{\epsilon_l}{2})] <= P[(p_i >= p_i^* + \frac{\epsilon_l}{2})|(p(a^*) >= p^*(a^*) - \frac{\epsilon_l}{2})]$$

Using Chernoff-Hoeffding bound and sample no. s definiton,

$$P[(p_i >= p_i^* + \frac{\epsilon_l}{2})|not(E_1)] <= e^{-2*\epsilon_l^2*s} <= \frac{\delta_l}{y}$$

Above is for one arm. Let b be the no of non-optimal arms whose estimated values are to be over estimated. Let's call them bad.Then,

$E[b|(p(a^*) >= p^*(a^*) - \frac{\epsilon_l}{2})]$ <= (No of bad arms)*(Probability of an arm being bad)

$$E[b|(p(a^*) >= p^*(a^*) - \frac{\epsilon_l}{2})] <= b * \frac{\delta_l}{y} <= n_l * \frac{\delta_l}{y}$$

No of non-optimal arms having estimated values more than that of optimal arms should be at least 3-4Th of total arms for eliminating the optimal one i.e. $b >= \frac{3n_l}{4}$. So, by **markov inequality**,

$$P[E_2] = P[b >= \frac{3n_l}{4}|not(E_1)] <= \frac{n_l\frac{\delta_l}{y}}{\frac{3n_l}{4}} = \frac{4\delta_l}{3y}$$

So, Total Failure probability = $P[E_1] + P[E_2] = \frac{\delta_l}{y} + \frac{4\delta_l}{3y}$. Being an (ϵ, δ)-PAC algorithm, it should be $<= \delta_l$.

$$\frac{\delta_l}{y} + \frac{4\delta_l}{3y} = \delta_l$$

This gives y = 7/3.

Thus, in a time-step, we need to sample each arm $\frac{2\ln(\frac{7}{3\delta_l})}{\epsilon_l^2}$ no of times.

1.3 ϵ and δ updates

Delta(δ):

δ has a basic condition that sum of all δ_l's must be $<= \delta$. At infinity, it should be equal to δ. Say $\delta_l = c^l * \delta$ where $0 < c < 1$

So, $\sum_{l=1}^{\infty} \delta_l = c * \delta + c^2 * \delta + ...$

$\sum_{l=1}^{\infty} \delta_l = \frac{c\delta}{1-c} = \delta$

This gives c = 1/2.

So, by this, we can write $\delta_l = \frac{\delta_{l-1}}{2}$ and $\delta_1 = \frac{\delta}{2}$

So, $\delta_l = \frac{\delta}{2^l}$.

Epsilon(ϵ):

ϵ has 2 conditions to satisfy;

1. $\sum \epsilon_l <= \epsilon$.
2. Convergence of series in sample complexity.

So, assuming $\epsilon_1 = \frac{\epsilon}{k}$ and $\epsilon_{l+1} = \epsilon_l \frac{k-1}{k}$ ensures that first condition that sum of ϵ_l's to be less than or equal to ϵ is satisfied, and we can write

$\epsilon_l = (\frac{k-1}{k})^{l-1} \frac{\epsilon}{k}$

1.4 Sample Complexity

For l^{th} round,

1. No of arms; $n_l = n(3/4)^{l-1}$ [Since, $n_l = 3n_{l-1}/4, n_1 = n$]
2. No of samples; $s_l = n_l * \frac{2}{\epsilon_l^2} \ln(\frac{7}{3\delta_l})$
3. $\epsilon_l = (\frac{k-1}{k})^{l-1} \frac{\epsilon}{k}$
4. $\delta_l = \frac{\delta}{2^l}$
5. Total no of rounds i.e. max value of l will be $\log_{4/3} n$

So, Sample complexity

$$\sum_{l=1}^{\log_{4/3} n} s_l = \sum_{l=1}^{\log_{4/3} n} n_l * \frac{2}{\epsilon_l^2} \ln(\frac{7}{3\delta_l})$$

Putting values of $\epsilon_l, \delta_l, n_l$ above and solving gives a term multiplied by

$$\sum_{l=1}^{\infty} (\frac{3}{4} * (\frac{k}{k-1})^2)^{l-1}$$

which has to be converged as summation is going till infinity and it will happen if $(\frac{3}{4} * (\frac{k}{k-1})^2)^{l-1} <= 1$ and this is satisfied at any value $k >= 2 + 4\sqrt{3}$. So, we will take k = 8.
So, $\epsilon_1 = \frac{\epsilon}{8}$ and $\epsilon_{l+1} = 7\frac{\epsilon_l}{8}$.

And simplifying the summation gives sample complexity as

$$O(\frac{n \ln(1/\delta)}{\epsilon^2})$$

1.5 Algorithm
Thus, the algorithm becomes as following:

Algorithm: Median Elimination (ϵ, δ)
1. Set S=A.
2. l=1,$\epsilon_1 = \frac{\epsilon}{8}$, $\delta_1 = \frac{\delta}{2}$
3. Sample every arm a in S for $\frac{2}{\epsilon_l^2} \ln(\frac{7}{3\delta_l})$ times and let p_a denote its empirical value.
4. Find the median of p_a. For the values lesser than p_a, find the median again and denote it by m_l.
5. $S_{l+1} = S_l - S_l\{a : p_a < m_l\}$
6. If $|S_l| = 1$, then output S_l. Else $\epsilon_{l+1} = \epsilon_l \frac{k-1}{k}, \delta_{l+1} = \frac{\delta_l}{2}$ and l = l+1. Go to 3.

2. (3 marks) 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 5 have 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?

Solution: Since, we know payoff values here, using them for better bounds make sense. We can use them to find regrets and include it in UCB1 which is ultimately called, improved UCB algorithm. In this, we first define $\Delta_i = q_(a_) - q_*(a_i)$. Including Δ_i in maximisation part gives

$$Q_j + \sqrt{\frac{2lnt}{n_j}}\Delta_j$$

So, we choose arm j which maximises this above term.

Now, finding the best action in minimum no of steps and then selecting the same action in a greedy way is the goal of a regret minimizing algorithm. If u_j part in UCB is lesser than 0.5*(highest payoff -2nd highest payoff) i.e.

$$u_j <= 0.5 * (4.6 - 3.1) = 0.75$$

Then it means we can distinguish the best action from the next best as we have ensured the low uncertainity(u_j). And then, we can say arm having highest estimate is the optimal arm. It can be done when u_j falls below 0.75.

3. (3 marks) Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 0.1 and 0.2 with probability 0.5 (case A), and 0.9 and 0.8 with probability 0.5 (case B).

   (a) (1 mark) If you are not able to tell which case you face at any step, what is the best expectation of success you can achieve and how should you behave to achieve it?

   Solution: If we don't know which case we will face at any step, then best way is to pick actions having best estimated true value.So, we can just take average true value of action a1 and a2 and take the maximum of the average true value of two actions.
   Average true value(a1) = 0.5*0.1 + 0.5*0.9 = 0.5
   Average true value(a2) = 0.5*0.2 + 0.5*0.8 = 0.5
   Since, both values are same, we can pick any action randomly.

   (b) (2 marks) Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

   Solution: Now, we know which case we are going to face, then we can ensure best expectation of success by picking action having higher true value in that case.
   In case A, a2 has higher true value. So, we will pick a2 in case A. Similarly, we will pick a1 in case B as it has higher true value.
   E[Success] = case A(0.5*True value of a2) + case B(0.5*True value of a1)
   = 0.5*0.2 + 0.5*0.9 = 0.55
   Thus best expectation of success is 0.55 by selecting action a2 in case A and a1 in case B.

4. (5 marks) Many tic-tac-toe positions appear different but are really the same because of symmetries.

   (a) (2 marks) How might we amend the learning process described above to take advantage of this? In what ways would this change improve the learning process?

   Solution: A tic-tac-toe game having just a CROSS or ZERO in a corner is same as having those in other 3 corners. So, folding the board to 1-4th of it won't make a difference on the game when symmetry is there. But it will lead to compact representation of states (Suppose we have already learnt about one state s1. Now ,in a game, we are on a state s2 which is symmetric of s1, so we can make our move using the information learnt in s1) and actions to be performed in that state. And thus, it will make the algorithm learn faster with lesser use of memory.

   (b) (1 mark) 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?

   Solution: If the opponent is not taking advantages of symmetry, it means it's playing correctly from one position and not from other symmetric ones. But in this case, if we will use symmetry i.e. treating all those symmetric positions as one, then we will fail to exploit that information. So, we can say that symmetrically equivalent positions don't always have the same value.

   (c) (2 marks) 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?

   Solution: If RL agent is playing against itself, it means a win for one will be a lose for other and vice-versa. If there is an option of draw, agents will try to draw the match as the no of runs go high. It is so because, winning and losing are same here as they are happening at same time. So, agents will eventually learn to draw effectively. Also, through this learning, it can learn better than what it could have learnt by playing against a random opponent.
   The agent will learn a different policy while playing against itself as it may not explore the non-optimal moves. So, when it encounters a random agent, when the opponent plays these non optimal moves,then agent may not know how to respond to these which it would have otherwise known had it been playing with a random opponent from start rather than playing with itself. Also, when

the agent is playing against itself, it won't be able to learn the biases of the opponent.

5. (1 mark) Ego-centric representations are based on an agent's current position in the world. In a sense the agent says, I don't care where I am, but I am only worried about the position of the objects in the world relative to me. You could think of the agent as being at the origin always. Comment on the suitability (advantages and disadvantages) of using an ego-centric representation in RL.

Solution: An ego-centric agent worries only about its immediate environment. It doesn't care about long term goals in a problem. It learns only when the immediate environment is new to it. So, if being in a similar situation as earlier, it will make the same move it learnt that time even if it will have negative effect on the game.
It can be good for immediate goals like avoiding immediate negative rewards and going towards positive immediate rewards i.e. it learns about favourable actions in immediate environment very quickly.

6. (2 marks) Consider a general MDP with a discount factor of γ. For this case assume that the horizon is infinite. Let π be a policy and V^π be the corresponding value function. Now suppose we have a new MDP where the only difference is that all rewards have a constant k added to them. Derive the new value function V^π_{new} in terms of V^π, c and γ.

Solution: So, we can write,
Value of a state s under policy π gives the average of discounted returns from rewards achieved on different paths. Policy π fixes which action to take with a certain probability, so, same action from state s can lead to different states and thus there is need of calculation of average of discounted returns.

$$V^\pi_{old}(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Now, all rewards have a constant k added to it. So, $r_{t(new)} = r_t + k$.

Therefore,
$V^\pi_{new}(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t(new)}] = E_\pi[\sum_{t=0}^{\infty} \gamma^t (r_t + k)]$

$= E_\pi[\sum_{t=0}^{\infty} \gamma^t r_t + \gamma^t k] = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_t] + E_\pi[\sum_{t=0}^{\infty} \gamma^t k]$

$= V^{\pi}_{old}(s) + \sum_{t=0}^{\infty} \gamma^t k = V^{\pi}_{old}(s) + k \sum_{t=0}^{\infty} \gamma^t$

So, from infinite sum of GP with first term as 1 and constant ratio as γ,

$$V^{\pi}_{new}(s) = V^{\pi}_{old}(s) + k * \frac{1}{1-\gamma} = V^{\pi}_{old}(s) + \frac{k}{1-\gamma}$$

7. (4 marks) An ϵ-soft policy for a MDP with state set $\mathcal{S}$ and action set $\mathcal{A}$ is any policy that satisfies

$$\forall a \in \mathcal{A}, \forall s \in \mathcal{S} : \pi(a|s) \geq \frac{\epsilon}{|\mathcal{A}|}$$

Design a stochastic gridworld where a deterministic policy will produce the same trajectories as a ϵ-soft policy in a deterministic gridworld. In other words, for every trajectory under the same policy, the probability of seeing it in each of the worlds is the same. By the same policy I mean that in the stochastic gridworld, you have a deterministic policy and in the deterministic gridworld, you use the same policy, except for ϵ fraction of the actions, which you choose uniformly randomly.

(a) (2 marks) Give the complete specification of the world.

Solution: We are seeing a transition from state s1 to s2 by taking action a. Having a ϵ-soft policy in a deterministic grid world makes probability of choosing action "a" in a particular state s1 as:$P[a|s_1] = (1-\epsilon+\frac{\epsilon}{|A|})$. Now, this action will lead to our desired state s2 as it's a deterministic world and we know choosing "a" leads to state s2.

For having same transition in a stochastic world having a deterministic policy, we need to make sure probability of transition from s1 to s2 to be as above. Since, it's a deterministic policy, we know in state s1, we always pick action "a" but we need to make sure that this action "a" leads to s2. So, probability for same can be written as: $P[s_2|(s_1, a)]$.

So,
$P[s_2|(s_1, a)]$ in stochastic world = $P[a|s_1]$ in deterministic world = $(1 - \epsilon + \frac{\epsilon}{|A|})$

This needs to be ensured to make sure that deterministic policy in a stochastic grid-world will give same trajectory as ϵ-soft policy in a deterministic world.

(b) (2 marks) Will SARSA on the two worlds converge to the same policy? Justify.

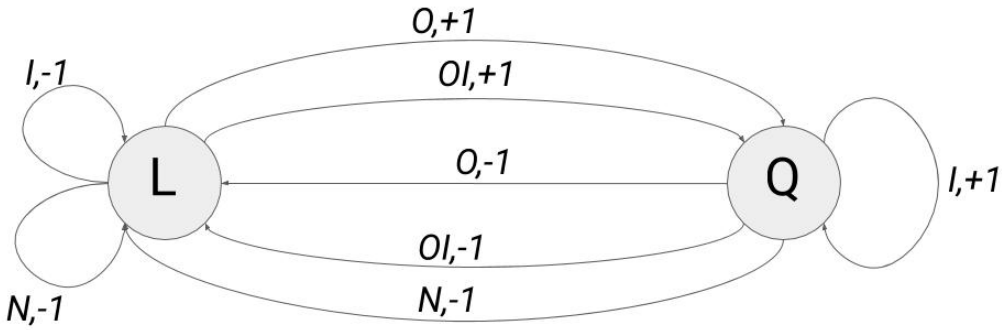
Solution:

8. (7 marks) You receive the following letter:
Dear Friend, Some time ago, I bought this old house, but found it to be haunted by ghostly sardonic laughter. As a result it is hardly habitable. There is hope, however, for by actual testing I have found that this haunting is subject to certain laws, obscure but infallible, and that the laughter can be affected by my playing the organ or burning incense. In each minute, the laughter occurs or not, it shows no degree. What it will do during the ensuing minute depends, in the following exact way, on what has been happening during the preceding minute: Whenever there is laughter, it will continue in the succeeding minute unless I play the organ, in which case it will stop. But continuing to play the organ does not keep the house quiet. I notice, however, that whenever I burn incense when the house is quiet and do not play the organ it remains quiet for the next minute. At this minute of writing, the laughter is going on. Please tell me what manipulations of incense and organ I should make to get that house quiet, and to keep it so.
Sincerely,
At Wits End

   (a) (3 marks) Formulate this problem as an MDP (for the sake of uniformity, formulate it as a continuing discounted problem, with $\gamma = 0.9$. Let the reward be +1 on any transition into the silent state, and -1 on any transition into the laughing state.) Explicitly give the state set, action sets, state transition, and reward function.

Solution: MDP Formulation:
State-space: {Laughter(L),Quiet(Q)}

Actions A: {Only Playing organ(O), Only burning incense(I), Doing Both(OI), Doing Nothing(N)}
State Transition Set: {R(L,O,Q), R(L,I,L), R(L,OI,Q), R(L,N,L), R(Q,O,L), R(Q,I,Q), R(Q,OI,L), R(Q,N,L)}
Rewards:
R(S1,A,S2) → Expected reward which agent gets when it performs action A in state S1 and reaches state S2.
Given : R(S,A,L) = -1 and R(S,A,Q) = +1



R(L,O,Q) = +1
R(L,I,L) = -1
R(L,OI,Q) = +1
R(L,N,L) = -1
R(Q,O,L) = -1
R(Q,I,Q) = +1
R(Q,OI,L) = -1
R(Q,N,L) = -1

(b) (2 marks) Starting with simple policy of **always** burning incense, and not playing organ, perform a couple of policy iterations.

Solution:

(c) (2 marks) Finally, what is your advice to "At Wits End"?

Solution: Our goal is to keep the house quiet. Since at the minute of writing, laughter is going on, so organ should be played for 1 minute which makes the house quiet and then after stopping organ, incense stick should be burned indefinitely. This will keep the house quiet forever.

9. (4 marks) Consider the task of controlling a system when the control actions are delayed. The control agent takes an action on observing the state at time t. The action is applied to the system at time $t + \tau$. The agent receives a reward at each time step.

(a) (2 marks)What is an appropriate notion of return for this task?

Solution: Since control actions are delayed, so, when the state s is observed at time t then actions are taken at t + τ and thus, rewards also are achieved at t + τ.
Say T = t + τ,
So, return can be written as:

$$G_t = R_{T+1} + \gamma^1 R_{T+2} + \gamma^2 R_{T+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{T+k+1}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+\tau+k+1}$$

(b) (2 marks) Give the TD(0) backup equation for estimating the value function of a given policy.

Solution: Say earlier, value function of a state s is V(s). Now, agent performs action a to move from s to $s^{'}$, then value function of state s based on this movement can be written as $R(s,a)+\gamma V(s^{'})$. So, temporal difference is defined as difference between these two i.e. TD = $(R(s,a) + \gamma V(s^{'}) - V(s))$. And α is a factor deciding how fast the algorithm learns. So, we update value function of s as:

$$V(s) \leftarrow V(s) + \alpha(R(s,a) + \gamma V(s^{'}) - V(s))$$

Here, $s = s_t$ $\quad s^{'} = s_{t+1}$ $\quad R(s,a) = R_{t+\tau+1}$
So,

$$V(s_t) \leftarrow V(s_t) + \alpha(R_{t+\tau+1} + \gamma V(s_{t+1}) - V(s_t))$$