CS6700 : Reinforcement Learning
Written Assignment #1

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

Name: Pragnesh Rana — **Roll number:** ME17S301

- 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:
For (ϵ, δ) PAC algorithm [1],
Let a' be an arm for which E[R(a')] < r^* - ϵ
$\Pr(r_{a'} > r_{a^*})$
$\leq P(r_{a'} > E[R(a')]) + \epsilon/2$ or $r^* - \epsilon/2$
$\leq P(r_{a'} > E[R(a')]) + \epsilon/2 + r^* - \epsilon/2$
$\leq 2\exp(-(\epsilon/2)^2 l)$
choosing the $l = \frac{4}{\epsilon}^2 \log \frac{2n}{\delta}$ assures that $\Pr(r_{a'} > r_a) < \delta/n)$

Now, The failure probability by $E = \{r_1^* < r_1 - \epsilon_1/2\}$
$\Pr\{r_j \geq r_1 | r_1 \geq -\epsilon/2\} \leq \Pr\{r_j \geq r_j + \epsilon/2 | r_1^* \geq r_1 - \epsilon/2\} \leq \frac{\delta_1}{3}$

Let $\#bad$ be the number of arms which are not $spdilon_1$-optimal but are empirically better than the best arm. $E[\#bad | r_1^* \geq r_1 - \epsilon/2] \leq n\delta_1/3$

$\Pr\{\#bad \geq n/4 | r_1^* \geq r_1 - \epsilon/2\} \leq \frac{n\delta_1/3}{n/4} = 4\delta/3$ which shows that prpobability of failure is bunded by δ

Number of samples in l^{th} round is $\frac{4\delta \ln(3/\delta_l)}{\epsilon_i^2}$

so,
$\delta_1 = \delta/2$; $\delta_l = \delta_{l-1}/2 = \delta/2l$
$n_1 = n/2$; $n_l = 3n_{l-1}/4 = (3/4)^{l-1}n$
$\epsilon_1 = \epsilon/2$; $\epsilon_l = 3/4\epsilon_{l-1} = (3/4)^{l-1}\epsilon/4$

Therefore,
$\sum_{l=1}^{\log_2 n} \frac{n_l \ln(3/\delta)}{(\epsilon_{l-1}/2)^2}$
$= 4\sum_{l=1}^{\log_2 n} \frac{(3/4)^{l-1} n \ln(2^l 3/\delta)}{((3/4)^{l-1}\epsilon/4)^2}$
$= 64\sum_{l=1}^{\log_2 n}(4/3)^{l-1}\frac{n}{\epsilon^2}\ln(2^l 3/\delta)$
As this series diverges so changing the epsilon,
$= \epsilon_1^2 = \epsilon^2/16$; $\epsilon_l^2 = (15/16)\epsilon_{l-1}^2 = (15/16)^{l-1}\epsilon^2/16$
so,
$= 4\sum_{l=1}^{\log_2 n} \frac{(3/4)^{l-1} n \ln(2^l 3/\delta)}{((15/16)^{l-1}\epsilon^2/16)}$
$= 64\sum_{l=1}^{\log_2 n}(4/5)^{l-1}\frac{n}{\epsilon^2}\ln(2^l 3/\delta)$
$= 64\frac{n}{\epsilon^2}\sum_{l=1}^{log_2 n}(4/5)^{l-1}[l\ln(2^l) + \ln(3) + \ln(1/\delta)]$
$\leq 64\frac{n}{\epsilon^2}\ln(1/\delta)\sum_{l=1}^{\infty}(4/5)^{l-1}[lC' + C]$
$\leq \mathcal{O}(\frac{n\ln(1/\delta)}{\epsilon^2})$

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:
1. yes, it is possible to design the regret minimizing algorithm toa achiee better bound than UCB.

2. **How UCB works?**
- A machine is randomly picked and Each machie have (uniform) same initial confidence interval and success distribution.
- Based on the reward, the confidence interval shifts either toward or away from the actual succesive distribution which eventually converges or due to exploration and results in the reductionn of the upper bound value of the confidence interval.
- Based on the updated upper cofidence boundof each machines, the one with highest is chosen to explore in the next round.

CHANGE: [2]
(LUCB++) : Rather than picking the best arm, it is prefreable to pick K-best arms. Finding the top-K arms required k and k+1 best arm, which makes confident that the worst arm in the top-K set is better than the best arm in the set of remaining arms.

3.**Modification in UCB:**
UCB only picks the best arm but in new algo K best are picked. By preceding in this manner by creating and refining estimates of the expected reward of the k-th and k+1-th best arm, termites the algo once the worst arm in the top k set is better than best arm in the set of remaining with desired confidence.

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: 1. As we don't know any detail priori, one cannot estimate individual detail for case A and B. The best approach to find out **best expectation such that which maximize the reward in combinations** for any case, means find optimal case such away that combination of both case is toward maximum reward and.
Possible cases are [3]:
$C_1 = 0.1 * 0.5 + 0.9 * 0.5 = 0.5$ **Best**
$C_2 = 0.1 * 0.5 + 0.8 * 0.5 = 0.45$

$C_4 = 0.2 * 0.5 + 0.8 * 0.5 = 0.5$ **due to possible dependency**
For given case maximum expected reward is 0.55. but out of first two case maximum expected reward is 0.5 and as action independence is not known so, C1 will preferred.

Second case is selected such away that which does not include any detail from case-1 as dependecy of actions not known.

To optimize reward in any case we prefer to select action in such way that selection of any case leads to maximize the for other case also.

For given case, estimate for two case is same in C1 and C4. The best expectation of success is 0.5 which can be achived by selecting action randomly at eah step.

$C_1 = 0.5 * 0.1 + 0.5 * 0.9 = 0.5$
$C_4 = 0.5 * 0.2 + 0.5 * 0.8 = 0.5$

(b) (2 marks) Now suppose that on each step you are told whether you are facing case A or case B (although you still dont 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: Associative search task involves trial and error learning in the form of search for the best arm.**??**. If we know we are either facing case A or case B then we can learn the best action by considering each case independent to each other.
so, possible combinations are,
$C_1 = 0.1 * 0.5 + 0.9 * 0.5 = 0.5$
$C_2 = 0.1 * 0.5 + 0.8 * 0.5 = 0.45$
$C_3 = 0.2 * 0.5 + 0.9 * 0.5 = 0.55$
$C_4 = 0.2 * 0.5 + 0.8 * 0.5 = 0.5$

Out of this possible cambination only that action is preferred which gives maximum reward which is C3,
$C_3 = 0.2 * 0.5 + 0.9 * 0.5 = 0.55$

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: In Tic-Tac-Toe, 4 axis of symmetry(two central axis and two diaonal axis) is possible. Using this 4 axis symmetry, the board can be fold down to a quarter of the original size.
Advantage of symmetry will reduce the memory requirement and increase the performance of the algorithm.

(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 opponent is not using advantage of symmetry then overall performance will become worse.

Example: Out of all positionn if opponent always played correct in one corner position denotes that player never took advantage symmetry. Which also shows that if you take advantage of symmetry then symmetrical equivalent position doesn't hold the same value in a multi-player game.

(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 player plays against itself then player will continue to adapt untill it reaches an equillibrium, which might be either fixed or cyclic.
After equillbrium it can learn higher skill against fixed opponent but it may happen that it may take wrong paths while reacting to itself.

5. (1 mark) Ego-centric representations are based on an agent's current position in the world. In a sense the agent says, I dont 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:

When agent is performing egonocentric leraning then it distinguishes situation based on it's immediate surrounding environment. For the case of cliff, learning Q-values where each state is set of immediate neighbours. [4]
Advantage:
Due to immediate neighbour **minimal number of death will be observed.**

Disadvatage:
- Due to immediate states number of success will be consistent or low as while avoiding cliffs it goes to other states randomly. - With increase in size of the area, wins reduces.

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:
It is given that
$v^{\pi}_{old}(s_i) = v^{\pi}(s_i) = \sum_{t=0}^{\infty} \gamma^t r_t$

but by adding constant to the reward obtain value fucntion is,

$v^{\pi}_{new}(s_i) = \sum_{t=0}^{\infty} \gamma^t (r_t + c)$
$v^{\pi}_{new}(s_i) = \sum_{t=0}^{\infty} \gamma^t r_t + \sum_{t=0}^{\infty} \gamma^t c$
$v^{\pi}_{new}(s_i) = v^{\pi}_{old}(s_i) + \sum_{t=0}^{\infty} \gamma^t c$

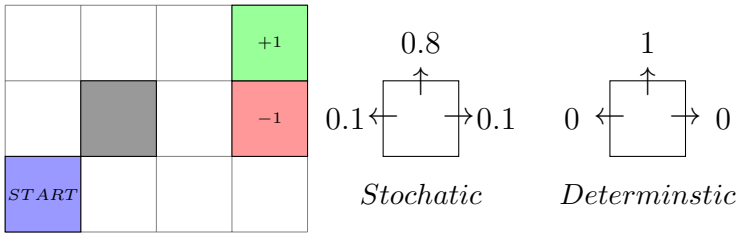
7. (4 marks) An ϵ-soft policy for a MDP with state set $\mathcal{S}$ and action set $\mathcal{A}$ is any policy that satisfies

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

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: Grid World:



In the maze like problem :
- agent lives in grid
- wall blocks agents path

With certain probability agent moves in certain direction(let's say with probability 0.8 agent moves in north direcion and with probability 0.1 agent moves in east and west direction)

Agents recives 'small'reward after each time step. After complition of successful attempt it receives the final reward ± 1.

Determinstic World: The set of states of the environment is composed of each cell of the grid. Set of action is composed of {left,right,up,down} and **given stat** $s \in S, \pi(s)$ **with probability 1 always takes same action for given policy.** If policy changes there might be chance of change of action.

Stochastic World : Set of action is composed of {left,right,up,down} and **given stat** $s \in S, \pi(s)$ **the agent move to next state with certain probability in next time-step .** In short, for same state and same policy there might be some chace that agent will select different state every next time.

For both cases, the agent uses determinstic policy.
- In stochastic grid world, determinstic policy is used so randomness in selection of state will be reduced and ultimately it picks the optimal path by updating the deterministic policy.

-In determinstic grid world, it also uses deterministic policy with ϵ-soft policy in which it follows the specific but ϵ time it randomly picked another state with uniform probability and try to update the policy and eventually converge to the final optimal policy which will be same above obtain policy and path.

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

Solution:
Yes, SARSA in the two world will converge to the same policy.[5]

SARSA(s,a,r,s',a') is on-policy reinforcement learning algorithm which estimates the value of policy being followed. It uses the action perfomed by the current policy to learn Q-value.

$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
SARSA accounts the current exploration policy which may be greedy with random steps.
Being greedy with random steps, it behaves like ϵ-greedy deterministic policy. It will find a policy that is optimal, taking into account the exploration inherent in the policy.

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:
State set: {L,Q} where,
L indicates laughter in the room
Q indicates Quite

Action set: $\{O \wedge I, O \wedge \neg I \, \neg O \wedge I, \neg O \wedge \neg I\}$
where,
O denotes to the playing the organ
I denotes burning incense
$\wedge$ denotes 'AND'
$\neg$ denotes 'Not'

Assuming determinstinc transition and reward following states are possible,

State Transition			Reward
Current state	Action	Next state	
L	$O \wedge I$	Q	+1
Q	$O \wedge I$	Q	+1
Q	$O \wedge \neg I$	L	-1
Q	$\neg O \wedge I$	Q	+1
Q	$\neg O \wedge \neg I$	L	-1

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

Solution:
policy :
$\pi\{a|$always burning incense , not playing organ$\}$

- if state = Laughter

  $\pi\{a|$always burning incense , play organ $\} \rightarrow$ state: Laughter $\rightarrow$ Quite

  if state = Quite

  $\pi\{a|$always burning incense , do not play organ $\}$ Quite $\rightarrow$ Quite

- if state = Quite

  $\pi\{a|$always burning incense , play organ $\} \rightarrow$ state : Quite $\rightarrow$ Noise

  if state = Quite

  $\pi\{a|$always burning incense , do not play organ $\}$ state: Quite $\rightarrow$ Quite

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

Solution:
It is advisable to play organ when there is laughter
and if room is quite, do not play the organ and burn incense.

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: For delayin control action and delayed reward, any approach which learns Q should work fine with delayed reward. Monte Carlo Control, SARSA, Q-learning, DQN and all other variants are in theory capable of learning the delayed reward.[6] **Return:**
New Return = old return +discounted factor * temporal difference
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+\tau}, a_{t+\tau} - Q(s_t, a_t))]$
generally **return in monte carlo** is obtained by,
$G_t = R_{t+\tau} + \gamma R_{t+2\tau} + ... = \sum_{k=1}^{\infty} \gamma^k R_{t+k\tau}$
where,
G_t = total discouter reward
γ = discount factor
$R_{t+\tau}$ = reward at time t+τ

which can be simplified as,
$G_t = R_{t+1} + \gamma R_{t+1} + ...$
Further
$G_t = R_{t+1} + \gamma G_{t+1}$
$G_t \approx R_{t+1} + \gamma v_(s_{t+1})$ [7]

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

Solution:
Value function is generally given as[7] ,
$v_{new}(s) = v_{old}(s) + \alpha[G_t - v_{old}(s)]$
now, replacing the target return using above obtained result,
policy can be obtain as,
$v_{new}(s_t) = vold(s_t) + \alpha[R_{t+1} + \gamma v_{old}(s_{t+1}) - v_{old}(s_t)]$

References

[1] E. Even-Dar, S. Mannor, and Y. Mansour, "Pac bounds for multi-armed bandit and markov decision processes," in International Conference on Computational Learning Theory, pp. 255–270, Springer, 2002.

[2] "Stochastic multi-armed bandits,pure exploration,"

[3] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. 2018.

[4] "Egocentric learning,"

[5] "On-policy learning,"

[6] "Delayed learning,"

[7] "Td(0),"