CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:55
Name: K Kamalesh Kumar **Roll number:** CE20B054

- 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) [Bandit Question] Consider an N-armed slot machine task, where the rewards for each arm a_i are usually generated by a stationary distribution with mean $Q^*(a_i)$. The machine is under repair when an arm is pulled, a small fraction, ϵ, of the times a random arm is activated. What is the expected payoff for pulling arm a_i in this faulty machine?

Solution: Let r_i denote the reward obtained on activating arm a_i. Then, the expected payoff for pulling a_i is

$$= (1-\epsilon)\mathbb{E}[r_i] \; + \; \frac{\epsilon}{N}\sum_{j=1}^{N}\mathbb{E}[r_j]$$

$$= (1-\epsilon)Q^*(a_i) \; + \; \frac{\epsilon}{N}\sum_{j=1}^{N}Q^*(a_j)$$

The above expression assumes that during the ϵ random selection, all arms are equiprobable.

2. (4 marks) [Delayed reward] Consider the task of controlling a system when the control actions are delayed. The control agent takes 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 an action A_t immediate reward occurs only at $t+\tau$, the usual notion of return is biased towards the choice of actions taken previously to A_t. A better formulation would be:

$$G_t = \sum_{i=0}^{T} \gamma^i \, r_{t+\tau+i}$$

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

Solution:

$$V_\pi(S_t) = V_\pi(S_t) + \alpha(R_{t+\tau} + \gamma V_\pi(S_{t+\tau}) - V_\pi(S_t))$$

3. (5 marks) [Reward Shaping] Consider two finite MDPs M_1, M_2 having the same state set, S, the same action set, A, and respective optimal action-value functions Q_1^*, Q_2^*. (For simplicity, assume all actions are possible in all states.) Suppose that the following is true for an arbitrary function $f : S \to R$:

$$Q_2^*(s,a) = Q_1^*(s,a) - f(s)$$

for all $s \in S$ and $a \in A$.

(a) (2 marks) Show mathematically that M_1 and M_2 has same optimal policies.

Solution: The optimal policy for M_1 is given by:

$$\begin{aligned}
\pi_1^*(s) &= \arg\max_a Q_1^*(s,a) \\
&= \arg\max_a Q_2^*(s,a) + f(s) \\
&= \arg\max_a Q_2^*(s,a) \\
&= \pi_2^*(s)
\end{aligned}$$

Hence, M_1 and M_2 have the same optimal policies.

(b) (3 marks) Now assume that M_1 and M_2 has the same state transition probabilities but different reward functions. Let $R_1(s,a,s')$ and $R_2(s,a,s')$ give the expected immediate reward for the transition from s to s' under action a in M_1 and M_2,

respectively. Given the optimal state-action value functions are related as given above, what is the relationship between the functions R_1 and R_2 ? That is, what is R_1 in terms of R_2 and f; OR R_2 in terms of R_1 and f.

Solution: Consider the Bellman optimality equation for Q_1^* and Q_2^*:

$$Q_1^*(s,a) = \sum_{s',r} p_1(s',r|s,a)\,[r + \gamma \max_a Q_1^*(s',a)]$$

Notice that,

1. $\sum_{s',r} p_1(s',r|s,a)\,[r] = \sum_{s'} p_1(s'|s,a) \overbrace{\sum_r p_1(r|s,a,s')[r]}^{R_1(s,a,s')} = \sum_{s'} p_1(s'|s,a) * R_1(s,a,s')$
2. $\sum_{s',r} p_1(s',r|s,a)\,[\gamma \max_a Q_1^*(s',a)] = \sum_{s'} p_1(s'|s,a)\,[\gamma \max_a Q_1^*(s',a)]$

Thus $Q_1^*(s,a) = \sum_{s'} p(s'|s,a)\,[R_1(s,a,s') + \gamma \max_a Q_1^*(s',a)]$ and similarly for Q_2^*. Here, $p(s'|s,a) = p_1(s'|s,a) = p_2(s'|s,a)$. On expanding the relation between Q_1^* and Q_2^*, we get:

$$\sum_{s'} p(s'|s,a)\,[\overbrace{R_2(s,a,s') + \gamma\,[\max_a Q_1^*(s',a) - f(s')]}^{unique}]$$

$$= \sum_{s'} p(s'|s,a)\,\overbrace{[R_1(s,a,s') + \gamma \max_a Q_1^*(s',a) - f(s)]}^{unique}$$

$$[\because \max_a Q_2^*(s',a) = \max_a Q_1^*(s',a) - f(s') \text{ and } f(s) = \sum_{s'} f(s) * p(s'|s,a)\,]$$

Since Q_1^* and Q_2^* are unique and the reward distributions are fixed, for a given s', the over braced terms are unique.

$$\therefore \;\; R_2(s,a,s') = R_1(s,a,s') + \gamma\, f(s') - f(s)$$

4. (10 marks) [Jack's Car Rental] Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited $ 10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of $ 2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number n

is $\frac{\lambda^n}{n!}e^{-\lambda}$, where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night.

(a) (4 marks) Formulate this as an MDP. What are the state and action sets? What is the reward function? Describe the transition probabilities (you can use a formula rather than a tabulation of them, but be as explicit as you can about the probabilities.) Give a definition of return and describe why it makes sense.

Solution: The MDP is formulated as the four tuple $\langle S, A, P, R \rangle$.

- The state space $S = S_1 \times S_2$, where $S_1 = \{1, 2, \cdots, 20\}$ and $S_2 = \{1, 2, \cdots, 20\}$. Simply put, it represents the number of cars at location-1 and location-2 at any given time-step.
- The action space $A(s)$ for a state s consists of the effective number of cars moved between locations. Or,
  $A(s) = \{1, 2, \cdots, \min(5, s_1 - 1), -1, -2, \cdots, -\min(5, s_2 - 1)\}$. Positive indicates moved from location $1 \to 2$ and vice-versa for negative numbers.
- Note that in this formulation, a state s is identified every morning (after the movement). Thus one timestep will elapse from one morning to the next.

Now, to formulate reward, consider x_1, x_2 to be the number of car requests at location 1 and 2. Similarly, the no. of car returns are denoted by y_1, y_2. All of them are Poisson random variables. Let the action taken be $a_t \in A(s_t)$. Then, the immediate reward and s_{t+1} are given by:

$$
\begin{aligned}
&r_t(s_t, a_t) = 10(x_1 + x_2) - 2|a_t| \quad \text{[which is total expenditure in \$]} \\
&s_{t+1,1} = s_{t,1} - x_1 + y_1 - a_t \\
&s_{t+1,2} = s_{t,2} - x_2 + y_2 + a_t \quad [s_{t+1} = (s_{t+1,1}, s_{t+1,2})\,]
\end{aligned}
$$

Coming to the transition dynamics P, it will simply be the probability of occurrence of all x_1, x_2, y_1, y_2 such that s_{t+1} and s_t are related as above, for a given a_t. If j belongs to the set of all such tuples, then

$$
\Pr(s_{t+1}|s_t, a_t) = \sum_{\forall j} \left(\frac{3^{x_1}}{x_1!}\mathrm{e}^{-3}\right)\left(\frac{4^{x_2}}{x_2!}\mathrm{e}^{-4}\right)\left(\frac{3^{y_1}}{y_1!}\mathrm{e}^{-3}\right)\left(\frac{2^{y_2}}{y_2!}\mathrm{e}^{-2}\right)
$$

The return G_t can be considered in the usual sense as future discounted rewards up till a horizon T. This allows Jack to consider sustained profit till T.

$$G_t = \sum_{i=0}^{T} \gamma^i r_{t+i}$$

(b) (3 marks) One of Jack's employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for free. Each additional car still costs \$ 2, as do all cars moved in the other direction. In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of \$ 4 must be incurred to use a second parking lot (independent of how many cars are kept there). These sorts of nonlinearities and arbitrary dynamics often occur in real problems and cannot easily be handled by optimization methods other than dynamic programming. Can you think of a way to incrementally change your MDP formulation above to account for these changes?

Solution: In this scenario, the action space $A(s)$ for a state s still remains the same, but for all the cars moved from location 1 to 2, one of them is driven for free by the employee. What changes is the reward dynamics. For cars moved from location 2 to 1, the reward is same as before, but for the vice-versa case, the reward becomes:

$$r_t(s_t, a_t) = 10(x_1 + x_2) - 2|a_t| + 2$$

The assumptions made here are, if he moves cars from location 1 to 2, at least one of them will be driven by the employee for free, and also, he does not move cars from both locations on the same night

(c) (3 marks) Describe how the task of Jack's Car Rental could be reformulated in terms of *afterstates*. Why, in terms of this specific task, would such a reformulation be likely to speed convergence? *(Hint:- Refer page 136-137 in RL book 2nd edition. You can also refer to the video at https://www.youtube.com/watch?v=w3wGvwi336I)*

Solution: In the original formulation, we considered the state to be determined every morning (after the movement of cars). That is, a timestep is considered from one morning to the next morning. But instead, if we consider the state to be determined the previous night, before the movement of cars, in which case a time step will be from one night to the next night, then it is easy to see that,

after the action (overnight moving of cars), and before the following morning, the augmented state can be considered as an $after\ state$.

It is possible to formulate this task as an after-state because the effect of the action on the state is deterministic, and the stochasticity only arises after the action is chosen.

This formulation helps because there could be many states with different numbers of cars in the two locations but could end up having the same number of cars after the action is chosen. So, multiple-state action pairs can be considered the same, removing the need to treat them separately. Thus, a value function can be learned over the after-states, which is efficient and would likely speedup convergence.

5. (8 marks) [Organ Playing] 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) (4 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: As before, the MDP can be described by $\langle S, A, P, R \rangle$

- The state space S consists of $\{laughing, silent, organ\ playing\}$. When the organ is playing, it is neither laughter or silence, thus it has been considered a separate state.

- The action space A is represented by $\{burn\ incense, play\ organ\}$
- State transitions are given by:
  - Laughing state, action is burn incense, next state is laughing
  - Laughing state, action is play organ, next state is organ playing
  - Silent state, action is burn incense, next state is silent
  - Silent state, action is play organ next state is organ playing
  - Organ playing state, action is play organ, next state is organ playing state
  - Organ playing state, action is burn incense, next state is silent state
- The reward function $r(s_t, a_t, s_{t+1})$ is as below:
  - $r(.\ ,.\ , laughing) = -1$
  - $r(.\ ,.\ , silent) = +1$
  - $r(laughing\ ,.\ , organ\ playing) = 0$ (Assumption)
  - $r(silent\ ,.\ , organ\ playing) = -1$ (Assumption)
- The return is given by $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$

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

Solution: Consider the value of all states to be initialized to zero. Performing dynamic programming:

$$V(laughing) = (-1 + 0.9 * 0) = -1$$
$$V(organ\ playing) = (1 + 0.9 * 0) = 1$$
$$V(silent) = (1 + 0.9 * 0) = 1$$

Doing one more loop of policy evaluation:

$$V(laughing) = (-1 - 0.9 * 1) = -1.9$$
$$V(organ\ playing) = (1 + 0.9 * 1) = 1.9$$
$$V(silent) = (1 + 0.9 * 1) = 1.9$$

Performing policy improvement (after finding the maximal action through one-

step look ahead):

$$\pi(laughing) = play\ organ$$
$$\pi(organ\ playing) = burn\ incense$$
$$\pi(silent) = burn\ incense$$

After one more step of policy iteration, it can be easily seen that the obtained policy is the same as above, and thus is the optimal policy

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

Solution: The advice would be to play the organ for one 1 min and then keep burning incense to continue in the silent state.

6. (4 marks) [Stochastic Gridworld] An ϵ-greedy version of a policy means that with probability 1-ϵ we follow the policy action and for the rest we uniformly pick an action. Design a stochastic gridworld where a deterministic policy will produce the same trajectories as a ϵ-greedy 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: Assumptions: For a given deterministic policy, the ϵ fraction of random actions taken in the deterministic grid world do not occur in the stochastic grid world. Thus the trajectories will not be the same. Hence only the state and rewards are considered in the trajectory, omitting the actions.

Design: If $s_t \in S$ is a state in both the worlds, all possible s_{t+1} in the deterministic gridworld, must also be possible in the stochastic gridworld. Consider a deterministic policy π with $\pi(s_t) = a_t$. In the deterministic grid world, the probability of observing s_{t+1}, r_{t+1} after s_t with the $\epsilon-$greedy policy is simply:

$$\Pr(s_{t+1}, r_{t+1}|s_t) = \overbrace{\pi_\epsilon(a_t|s_t)}^{\epsilon-version\ of\ \pi} = 1 - \epsilon + \frac{\epsilon}{\mathrm{N}(A(s_t))}$$

For such a trajectory to be equally probable in the stochastic gridworld following

π, it follows that the dynamics function satisfies:

$$p(s_{t+1}, r_{t+1}|s_t, a_t) = 1 - \epsilon + \frac{\epsilon}{\mathrm{N}(A(s_t))}$$

The above is when a_t is the deterministic action. To fully specify p for a given s_t:

$$\forall\, a \in A(s_t) \Rightarrow p(s, r|s_t, a) = 1 - \epsilon + \frac{\epsilon}{\mathrm{N}(A(s_t))}$$

where s, r is the transition that occurs in the deterministic gridworld, when a is taken in s_t

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

Solution: SARSA should converge to the same optimal policy in both gridworlds. Consider the TD target at time-step t:

$$\mathrm{TD}_{target} = r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1})$$

If the action a_t is the greedy action for the deterministic gridworld, it is very likely that it is also the greedy action for the stochastic gridworld, since s_{t+1} for both the worlds is the same with high probability. In other words, a_t also maximizes the expected return for the stochastic world.

It is easy to see the same if we initialize the same Q values for both worlds, and perform SARSA. Obviously, to converge to the same optimal policy, it may be required ϵ and α are decayed alike for both worlds.

7. (5 marks) [Contextual Bandits] Consider the standard multi-class classification task (Here, the goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs). Can we formulate this as a contextual bandit problem (Multi armed Bandits with side information) instead of standard supervised learning setting? What are the pros/cons over the supervised learning method. Justify your answer. Also describe the complete Contextual Bandit formulation.

Solution: Yes, it can be formulated as a contextual bandit problem. One such formulation is outlined here. Consider a bandit with the number of arms as the number of classes. So, pulling an arm would be the same as classifying a data point. Additionally, while choosing an arm, a feature vector corresponding to the data point

is also considered. A numerical preference for each arm for a given data point can be formulated, and a softmax over them will represent how likely is each arm (class) for the datapoint:

$$\Pr(A_t = a; X_t = x) = \frac{e^{H_\theta(a;x)}}{\sum_{\forall a} e^{H_\theta(a;x)}} = \pi_\theta(a; x)$$

Here, H is parameterized through θ, for e.g, like a neural network. At a given timestep, all the datapoints can be passed to the bandit, and the expected reward (or performance) can be approximated as below:

$$\mathcal{J}_\theta = \mathbb{E}(R_t) \approx \sum_{i=1}^{N} \sum_{\forall a} \pi_\theta(a; x_i) R(a, y_i)$$

where $R(a, y_i)$ is some handcrafted reward function which considers x_i was classfied as a when true class was y_i. It could be as simple as $+1$ for correct classification and 0 otherwise, or it could also be some common performance metrics computed over the whole dataset. Finally, to obtain the optimal classifier π_θ, we perform gradient ascent as below:

$$\theta_{t+1} = \theta_t + \eta \nabla \mathcal{J}_{\theta_t}$$

During test-time a prediction can be made as below:

$$\hat{y} = \arg\max_a \ \pi_{\theta_*}(a; x)$$

Pros/Cons:

One of the primary advantages of contextual bandit formulation is the flexibility of choosing a reward function. The reward function can also be non-differentiable as is the case above, whereas the equivalent "loss functions" in the supervised setting require it be differentiable so as to backpropagate the gradients as in neural networks.

On the other hand, the bandit setting could face convergence and high-variance issues and may not be scalable to larger problems. The high variance could still be tackled by adopting baselines.

8. (5 marks) [TD, MC, PG] Suppose that the system that you are trying to learn about (estimation or control) is not perfectly Markov. Comment on the suitability of using different solution approaches for such a task, namely, Temporal Difference learning, Monte Carlo methods, and Policy Gradient algorithms. Explicitly state any assumptions that you are making.

Solution: Assuming there is only a finite amount of experience in terms of episodes, wherein policy evaluation can iterated over the same experience in a batch fashion, MC-based methods are more suitable when the environment is not exactly Markov. In MC, the value functions converge to the actual average returns observed in the seen data. In other words, "batch-MC" minimizes the mean squared error on the data on which it was evaluated.

On the other hand, TD-based methods assume Markovness in the underlying process and implicitly create a maximum-likelihood Markovian model that best explains the data. In other words, the value function that "batch-TD" converges is that of the Markovian model of the seen data, and such an estimate has been called the *certainity equivalence estimate.* Since TD makes such a Markov assumption, it may not be suitable when the process is not fully Markov. Policy gradient methods for control also make the Markov assumption while modeling the policy, but Monte Carlo-based policy gradient methods could be more suitable than PG methods that depend on TD.

9. (5 marks) [PG] Recent advances in computational learning theory, have led to the development of very powerful classification engines. One way to take advantage of these classifiers is to turn the reinforcement learning problem into a classification problem. Here the policy is treated as a labeling on the states, and a suitable classifier is trained to learn the labels from a few samples. Once the policy is adequately represented, it can be then used in a policy evaluation stage. Can this method be considered a policy gradient method? Justify your answer. Describe a complete method that generates appropriate targets for the classifier.

Solution: One approach to using classifiers for updating the policy would be to set the actions in a state as different classes and classify a state into that action that has the maximal action value for that state. So the ground-truth label for a state will be the greedy action for the current action values for the state, where the action values are evaluated for the current policy (policy evaluation). Now, fitting a classifier to predict these greedy actions given the state as input will be equivalent to doing a policy improvement. This whole process can once again be seen as Generalized Policy Iteration (GPI).

To make it more formal, consider a policy π and action-value function $Q_\pi(s, \ .\ )$ for a state a, which could have been estimated for policy π through Monte Carlo or TD evaluation. Let a_* be the maximizing action:

$$a_* = \arg\max_a Q_\pi(s, a)$$

A datapoint for the classifier can be created as (s, a_*), where s is the input to the classifier and a_* is the label to s. Similarly, multiple datapoints can be created, ensuring that the collective samples are representative of the state space. Thus the dataset for the current policy π is represented by:

$$\mathcal{D}_\pi = \{(s_1, a_{1*}), (s_2, a_{2*}), \cdots, (s_n, a_{n*})\}$$

A classifier $\mathbb{C}$ can be trained on these samples to predict the greedy action for a state input. After this round of policy improvement, the improved policy π_{imp} is given by:

$$\pi_{imp} = \mathbb{C}(\mathcal{D}_\pi)$$

Once again, policy evaluation can be carried out on π_{imp}, and a new dataset $\mathcal{D}_{\pi_{imp}}$ can be created to train the classifier for further policy improvement. A typical classifier could be a neural network.

It is not appropriate to consider this approach as a *policy gradient method* as there is no explicit performance measure (wrt rewards) that is being maximized, even though the policy is parameterized by parameters θ of the classifier whose loss function is minimized through gradient descent (eg: in a neural network).