AI 3000 / CS 5500 : Reinforcement Learning
Exam № 2

Due Date : 07/11/2021, 12.30 PM

Easswar Subramanian, IIT Hyderabad 07/11/2021

Instructions

Read all the instructions below carefully before you start answering the questions

- Please include your institute roll number in the **first** page of the answer sheet
- This is an open-book exam.
- Seeking help from other individuals (including classmates) is **not** allowed.
- **Plagiarism in answers will be dealt with strictly**.
- The exam has 4 problems for a total of 50 points.
- All answers should include **suitable justification** lest no marks will be awarded.
- The estimated amount of work for this exam is about three hours.
- Please submit the answer sheets by **12:30pm IST, Tuesday December 07, 2021**.
- Submit your answer-sheet as a **private post to me and the Instructors** on Piazza under the **midterm-exam** tab.

Problem 1 : Markov Decision Process, Dynamic Programming

Consider a MDP $M < \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >)$ with 2 states $\mathcal{S} = \{s_1, s_2\}$. From each state there are 2 available actions $\mathcal{A} = \{a_1, a_2\}$. Choosing action a_1 from any state leaves you in same state and gives a reward of -1. Choosing action a_2 from state s_1 takes you to state s_2 by giving a reward -2, while choosing a_2 from state S_2 ends the episode giving reward 3.

(a) Depict the above MDP in figure. (1 Point)

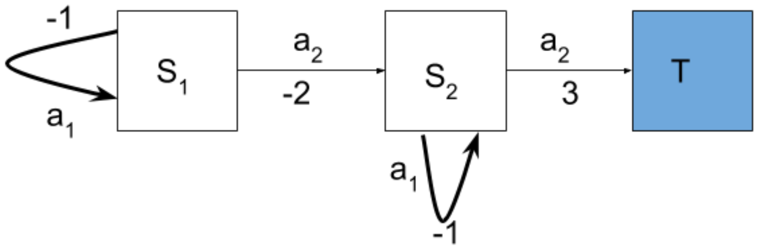

Figure 1: Markov Decision Process

It is useful to consider a <u>notional</u> terminal state like T, to represent episode termination, as that it will aid in value iteration algorithm. .

[Some of you have drawn action a_2 ending in state s_2 with reward 3 and that is wrong and it will also not help you in correct value function computation.]

(b) Initialize value function to zero and perform value iteration (for maximum of 5 iterations) and comment whether the value function converges to optimal value function. (2 Points)

Let us initialize value iteration as $V_0 = [0; 0]$. Then $V_1 = [-1; 3]$ and $V_2 = [1; 3]$. We also have $V^* = [1; 3]$. It only takes two iterations for the value function to converge.

(c) Is the convergence or successive value function iterates monotonic for all states of the MDP ? If no, would that contradict the fact the Bellman optimality operator is a contraction ? Explain. (2 Points)

Clearly, for state S_1, convergence is clearly not monotonic. The Bellman operator is a contraction. This means that the maximum absolute value across all states of the error (difference of the value function compared to the optimal value function) must not increase between iterations of value iteration. However, for individual states it is possible that the error increases between iterations. In this example, $\|V_0 - V^*\|_\infty = 3, \quad \|V_1 - V^*\|_\infty = 2, \quad \|V_2 - V^*\|_\infty = 0$. So clearly there is no contradiction. The key idea is here is the use of infinity or max norm, so considering error estimates of value function on individual states is not the correct way to verify the contraction property.

(d) Prove that a discounted MDP with finite state and finite actions with bounded rewards have bounded returns (2 Points)

Without loss of generality, we can assume that each reward r_t of the MDP is non-negative and be upper bounded by R_{max}. Since each reward is non-negative, we have,

$$0 \leq \sum_{t=1}^{\infty} \gamma^{t-1} r_t \leq \sum_{t=1}^{\infty} \gamma^{t-1} R_{max} = \frac{R_{max}}{1-\gamma}$$

Hence, the discounted sum of rewards (or the discounted return) along any actual trajectory is always bounded in range $[0, \frac{R_{max}}{1-\gamma}]$ and so its expectation of any form. When the rewards may be negative, but is still in the bounded range $[-a, b]$ with $a, b > 0$, we can add a constant offset $c = a$ and let $R_{max} = a + b$

(e) Suppose that we are solving a shortest path problem using an MDP. What is the suitable choice of discount factor of the MDP ? Explain. (1 Point)

The discount factor for SSP MDP can be set to 1. We are anyway considering shortest path problems and it is often the case we consider proper trajectories (trajectories that always terminate in a goal state with probablity 1). For these kind of MDPs, $\gamma = 1$ is a suitable choice.

Problem 2 : Q-Learning and Function Approximation

(a) Consider the following update rule of the (Watkin's) tabular Q-learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \max_{a'} \gamma Q(s_{t+1}, a') - Q(s_t, a_t) \right).$$

The online or incremental Q-learning algorithm, explained in Lecture 15 (DQN networks, slide 10) also had exactly the same target, except that it uses a function approximator. Why does then, only the online Q-learning algorithm suffers from the moving target problem and not the (Watkin's) tabular Q-learning algorithm ? (2 Points)

Online Q-learning suffers from moving target problem because it uses a function approximator with parameter ϕ which is updated in every step and hence the target keeps changing at every step. Watkin's Q-learning does not have any parameters and hence do not suffer from moving target problem.

(b) We are given a stream of n experience quadruples denoted by (s_i, a_i, r_i, s_i'). The goal is to learn an optimal (action) value function using a function approximator. Consider the following approach to find $Q_\phi(s, a)$, that involves, performing stochastic gradient descent for each batch of k quadruples (s_i, a_i, r_i, s_i') with the loss function given by $\sum_{i=1}^{K} \left(Q_\phi(s_i, a_i) - y_i\right)^2$ where $y_i = r_i + \max_{a'} \gamma Q_\phi(s_i', a')$. What do you think is the disadvantage of this approach ? Is the above method of learning the Q network guaranteed to give us an approximation of the optimal state action value function ? (3 Points)

Provided the stream of n experiences are not sequentially correlated, it is OK to perform the regression suggested. But it can suffer from moving target problem, provided the K is not large enough. Convergence to optimal Q function is anyway not guaranteed due to the fact that composiiton of projection and Bellman backup operator is not contraction.

(c) The optimal Q function $Q^*(s,a)$ is exactly representable by some neural network architecture $\mathcal{N}$. Suppose we perform Q-learning on a MDP using the architecture $\mathcal{N}$ to represent the Q-values. Suppose we randomly initialize the weights of a neural net $\mathcal{N}$ and collect infinitely many samples using an exploration strategy that is greedy in the limit of infinite exploration (GLIE). Are we guaranteed to converge to the optimal Q function $Q^*(s,a)$? Explain your answer. (2 points)

Because this method of Q-learning involves function approximation, bootstraping and off-policy training, we are not guaranteed to converge to anything.

(d) What is maximization bias ? How does the Q-learning update rule gets affected by maximization bias ? How does Double Q-learning update rule alleviate this bias ? (3 points)

Maximization Bias : Maximization bias is due to Jensen's inequality that can be stated as follows. Let X_1 and X_2 be two random variables. Then

$$\mathbb{E}(\max(X_1, X_2)) \geq \max(\mathbb{E}(X_1), \mathbb{E}(X_2)).$$

In the context of Q-learning, this inequality can result in positive bias while choosing (optimal) actions and their Q-values (for target computation) based on finite sample estimates. This is called Maximization bias.

- The problem with vanilla tabular Q-Learning is that the same samples are being used to decide which action is the best (highest expected reward), and the same samples are also being used to estimate that action-value
- Break up the Q-learning update rule as follows

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left(r_{t+1} + \gamma Q(s_{t+1}, \arg\max Q(s_{t+1}, a)) - Q(s_t, a)\right)$$

- If action value is overestimated, then it is chosen as the best action, and its overestimated value is used as the target (this results in maximization bias)
- **Solution in Double Q learning** : Have two different set of samples to decide the action and to evaluate the target
- The idea is to use two Q functions
- In the tabular Q-learning setting, for each transition quadruple $(s_t, a_t, r_{t+1}, s_{t+1})$ we flip a fair coin to decide any of the two update steps given below,

$$Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q_2(s_{t+1}, \arg\max Q_1(s_{t+1}, a)) - Q_1(s_t, a_t)\right)$$

$$Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q_1(s_{t+1}, \arg\max Q_2(s_{t+1}, a)) - Q_2(s_t, a_t)\right)$$

Problem 3 : Policy Gradients

(a) Let $\pi_\theta(a|s)$ denote a parameterized policy used in the policy gradient set up. Consider a discrete action space MDP, wherein a softmax policy is used for choosing actions, defined as follows,

$$\pi_\theta(a|s) = \frac{e^{\phi(s,a)^\intercal\theta}}{\sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta}}.$$

Derive an experssion for the score of this softmax policy. (3 Points)

$$\log(\pi_\theta(s,a)) = \log(e^{\phi(s,a)^\intercal\theta}) - \log(\sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta})$$

Now take the gradient,

$$\nabla_\theta \log(\pi_\theta(a|s)) = \nabla_\theta \log(e^{\phi(s,a)^\intercal\theta}) - \nabla_\theta \log(\sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta})$$

The first term simplifies to $\phi(s,a)$ and for the second term use the log-derivative trick

$$\nabla_x \log(f(x)) = \frac{\nabla_x f(x)}{f(x)}$$

The second term simplifies to,

$$\nabla_\theta \log(\sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta}) = \frac{\nabla_\theta \sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta}}{\sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta}}$$

Taking the gradient of the numerator, we get,

$$\frac{\sum_{k=1}^N \phi(s,a_k) e^{\phi(s,a_k)^\intercal\theta}}{\sum_{k=1}^N e^{\phi(s,a_k)^\intercal\theta}}$$

which then leads to the following expression,

$$\nabla_\theta \log(\pi_\theta(a|s)) = \phi(s,a) - \mathrm{E}_{\pi_\theta}[\phi(s,\cdot)]$$

(b) Now consider a continuous action space MDP and a Gaussian policy to sample action a from a normal distribution with mean $\phi(s)^T\theta$ and variance σ. Derive an expression for $\nabla_\theta \log(\pi_\theta(a|s))$ and $\nabla_\sigma \log(\pi_\theta(a|s))$ (6 points)

As a reminder, the Gaussian PDF is as follows,

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Write down the log density as,

$$\log \pi_\theta(a|s) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\left(\frac{\phi(s)^T\theta - a}{\sigma}\right)^2$$

Differentiate w.r.t to θ, and after simplification, we have,

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{a - \phi(s)^T\theta}{\sigma^2}\phi(s)$$

Differentiating log density w.r.to σ, and after simplification, we have,

$$\nabla_\sigma \log \pi_\theta(a|s) = \frac{(\phi(s)^T\theta - a)^2}{\sigma^3} - \frac{1}{\sigma}$$

(c) What are the advantages of using the REINFORCE algorithm over DQN for solving the RL problem ? What are the disadvantages of using the REINFORCE algorithm, if any, over DQN. (2 Points)

REINFORCE algorithm advantages ; (a) Suitable for continuous action space. (b) Better convergence properties than DQN (as it just gradient descent). Disadvantage of REINFORCE is that (a) it is an on-policy algorithm whereas DQN is off-policy algorithm; (b) REINFORCE is Monte Carlo based and requires full trajectory sequence to compute gradient updates while DQN is an online algorithm.

(d) The policy gradient theorem provides an expression for the gradient of the performance measure of a parameterized policy π_θ as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\Psi_t\right].$$

One popular expression for Ψ_t is given by, $\Psi_t = \gamma^t\left[r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right]$ which is an approximation to the advantage function. Using the definition of advantage function, justify the above formulation of Ψ_t. (2 Points)

Consider the definition of advantage function

$$A^\pi(s,a) \overset{\text{def}}{=} Q^\pi(s,a) - V^\pi(s)$$

. Now, consider one-step TD error for V^{π_θ}

$$\delta_t^{\pi_\theta} = r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$$

. Upon taking expectation, we have,

$$\begin{aligned}\mathbb{E}_{\pi_\theta}(\delta^{\pi_\theta}|s_t,a_t) &= E_{\pi_\theta}(r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1})|s_t,a_t) - V^{\pi_\theta}(s_t)\\ &= Q^{\pi_\theta}(s_t,a_t) - V^{\pi_\theta}(s_t) = A^{\pi_\theta}(s_t,a_t)\end{aligned}$$

Hence, the one-step TD error is an unbiased estimate of the advantage function. In practice, we use the approximate one step TD error given by

$$A^{\pi_\theta}(s_t,a_t) \approx r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

where V_ϕ is the function approximator for V^{π_θ}.

(e) Let us consider an alternative expression for Ψ_t given by $\Psi_t = \sum_{t'=t}^{\infty} \gamma^{t'} r_{t'+1} - b_t$ where b_t is a constant or a time-dependent baseline. What are the advantages and disadvantages of this formulation over the formulation of Ψ_t in sub-question (b) ? (2 Points)

The expression $\Psi_t = \sum_{t'=t}^{\infty} \gamma^{t'} r_{t'+1} - b_t)$ is Monte Carlo based advantage estimate. MC based expression requires whole trajectory sequence to compute gradient update. Second, Monte Carlo estimates have high variance, as they are estimates from a single trajectory. But the upside of the MC based estimates is that they are unbiased in expectation. Also, using this estimate for PG methods does not introduce moving target problem.

(f) How would you compute the expression $\Psi_t = \gamma^t \left[r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right]$ while implementing a policy gradient algorithm for an RL problem with continuous state space ? What issues might arise in such implementation ? What are the alternate ways to reformulate the advantage function to suitably address such issues ? (2 Points)

Although in theory, the expression $\Psi_t = \gamma^t \left[r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right]$ is an unbiased estimator of the advantage function, in practice, we use the following approximation

$$A^{\pi_\theta}(s_t, a_t) \approx r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

which introduces bias because we now use function approximator V_ϕ for V^{π_θ}. Also, straightforward implementation of the above expression leads to mvoing target problem. The moving target target in PG methods is alleviated either by using Batch algorithms or A3C type algorithms. The bias-variance trade-off can be handled better one use multi-step TD errors and TD-λ based advantage estimators.

(g) The following objective function is the starting point for many policy gradient formulations

$$J(\theta) = V^{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right].$$

Here $s_0 \in \mu(s)$ is a start state for trajectories sampled using policy π_θ. What approximations to the objective function $J(\theta)$ does one deploy to derive the expression for natural gradients ? Under what conditions such approximations are valid ? (3 Points)

First $J(\theta)$ is approximated by a surrogate loss function. The surrogate function approximation is valid only two consecutive policy of the PG method are close in terms of the KL divergence metric. This leads to a constraint optimization problem with a KL penalty term. Then, further approximation is done by converting an optimization problem with KL penalty term into a constrained optimization problem with KL constraint. Thereafter, to get analytical solutions, we linearize the objective and quadratify the constraint using Taylor series.

Problem 4 : Multi-Arm Bandits

(a) What problems does the Multi-arm bandit (MAB) framework solve ? (1 Point)

The MAB framework solve problems that involves one step decision making (across time to minimize regret).

(b) What is the difference between Multi-arm bandit (MAB) setting and the full RL setting ? (2 Points)

The bandit setting involves one step decision (across time to minimize regret) whereas the full RL formulation involves sequential decision making (across multiple time steps) to optimize an objective. This is the reason there are no state transitions in MAB setting. Nevertheless, one could have multiple states even in a bandit setting (for example, contextual bandits).

(c) What is optimism in the face of uncertainty and how is it different from naive exploration approach ? (2 Points)

Naive exploration involves exploring arms (or actions) without any strategy (like uniform random). Whereas, optimism in the face of uncertainty involves picking an arm (or action) that we are most uncertain about (like least picked action so far).

(d) Which step of the UCB1 algorithm uses the optimism in the face of uncertainty principle ? And how ? (2 Points)

The UCB1 expression provided in the lecture slide has two components. The second term containing the number of times an arm of pulled in comparison to the total number of trials is where optimism in the face of uncertainty is used in UCB1 algorithm.

(e) After 12 iterations of UCB1 algorithm applied on a 4-arm bandit problem, we have $n_1 = 3, n_2 = 4, n_3 = 3, n_2 = 2$ and $Q_1 = 0.55, Q_2 = 0.63, Q_3 = 0.61, Q_4 = 0.4$. Which arm would be played next ? (2 Points)

Calculating the argmax coffecients using the above values using the above data, we have, $U_1 = 1.837, U_2 = 1.745, U_3 = 1.897, U_4 = 1.976$, Clearly, arm 4 has the highest upper confidence bound and hence will be selected by the UCB1 algorithm.

(f) What is the underlying principle of the Thompson sampling algorithm that differentiates it from algorithms with UCB flavour ? (1 Point)

Thompson sampling is a Bayesian approach to solve MAB problems by computing posteriors from priors. Whereas UCB1 is a frequentist approach which doesn't have any assumptions on reward distributions of the arms.

(g) Which step of the Thompson algorithm ensures that one performs exploration ? Explain. (2 Points)

At each step, we sample the expected rewards of each arm from their corresponding reward distributions. This introduces an element of exploration since even an arm for which the mean of the distribution is less than the mean of the distribution of another arm can be selected by the algorithm if the expected reward sampled for the former arm is larger than the expected reward sampled for the latter arm. (Step No. 4 of the Thompson algorithm slide from the lecture.)

ALL THE BEST