Mahindra University Hyderabad
École Centrale School of Engineering
Mock Paper for End Semester Exam

Program: B. Tech. Branch: AI/CSE Year: 4 Semester: 1
Subject: Reinforcement Learning and Autonomous Systems (AI 4102)

Time Duration: 3 hours **Max. Marks: 100**

Instructions:

1) The submitted answer sheet should only contain your final answer. Use separate sheets for rough work.
2) You must show your solution procedure in the submitted answer sheet.

Q1: Bandits and Reinforcement Learning **(20 marks)**

(a)

Environment 1	**Environment 2**
$V^*(x) = \max_{a \in A(x)} \left(r(x,a) + \sum_{x' \in S} \theta_{x,x'} V^*(x') \right)$	$V^*(x) = \max_{a \in A(x)} \left(r(x,a) + \sum_{x' \in S} \phi_{x'} V^*(x') \right)$
where $\theta_{x,x'}$ is the transition probability from state x to x' in two consecutive time slots.	where ϕ_x is the probability of state x.

Consider two different environments with the **same** *(i)* state, $\boldsymbol{x}$, *(ii)* state space, $\boldsymbol{S}$, *(iii)* action, $\boldsymbol{a}$, *(iv)* action space, $\boldsymbol{A(x)}$, and *(v)* average reward for state-action pair, $\boldsymbol{r(x,a)}$. The **only difference** is in there **transition probability** distribution. The Bellman optimality equations for these two environments are shown in the above table.

For each of these two environments, comment if it is a Multi-Armed Bandit, Contextual Bandit, or Markov Decision Process. You must justify your answer. **(6 marks)**

PLEASE NOTE: We can always write Bellman optimality equation for Bandit setup as well because the are **special cases** of Markov Decision Process.

(b) There is an policy for Multi-Armed Bandits (MAB) called "median elimination" that was NOT taught during lecture. Your task is to read the following description of a **simplified** version of median elimination policy and then convert the description into a psuedocode.

Median elimination is a **value-based** policy for MAB (NOT contextual bandits). Since it is a value-based policy, it keeps an estimate of the average reward of each action a prior to time t, $Q_t(a)$. The main difference (from what you learned in lecture) is that in median elimination, the time horizon is divided into "rounds". Each round is of τ **consecutive** time slots, where $\tau > 1$. In every round, the agent keeps a set, $\boldsymbol{S}$, that contains those actions that can potentially be the one with the highest (true) mean reward. For the first round, all the actions are there in $\boldsymbol{S}$. In the **beginning** of subsequent rounds, the agent does the following:

1. Computes the median of $Q_t(a)$ for all $a \in S$.

2. Remove all those actions, a, from S whose $Q_t(a)$ is less than the median.

The **termination criteria** is that S should contain only one action. Write a psuedocode for median elimination. While writing this psuedocode, you can assume that you are **given a function that can sort an array**. **(10 marks)**

(c) In the median elimination policy described above, τ is a hyperparameter. Now, consider the following two cases:

Case 1: For every action, the mean reward is between 20 to 35 and the standard deviation is between 5 to 30.

Case 2: For every action, the mean reward is between 10 to 25 and the standard deviation is between 15 to 40.

Suppose that the agent knows the range of mean and standard deviation for both the cases (the agent DOES NOT know the exact mean and standard deviation for each arm; just the range). For which of the two cases, will the value of τ be higher. Justify your answer. **(4 marks)**

Q2: Markov Decision Process **(20 marks)**

In cloud computing, we buy virtual machines, sometimes called "instances", in order to do our computational task. Cloud computing service like Amazon Web services have two kind of instances: "on-demand instances" and "reserved instances". On demand instances can be bought on for an hour and are hence good for "short-lived" computational demand . Reserved instances can be bought for around six months. They are prefered for "long-lived" computational demand as we get "bulk discount" when we buy them.

We consider that the time is divided into slots. The duration of a time slot is one hour (the duration of an on-demand instance). The computational demand at time t is d_t units where $d_t \in \{0,1,2,\dots,D\}$. The transition probability from demand d to d' in two successive time slots is

$\theta_{d,d'}$. An agent can **buy** and **use BOTH** on-demand and reserved instances in every time slot. Each on-demand instance and reserved instance can be used to served M units of customer demand. However, these two instance types differ in the following ways:

1. *On-demand instance*: The cost of buying an on-demand instance is p rupees. An on-demand instance bought at time t, can be used to serve the computational demand at time t only after which the instance expires.
2. *Reserved instance*: The cost of buying a reserved instance is P rupees. A reserved instance once bought lasts for $\tau > 1$ time slots. In other words, a reserved instance bought at time t, can be used to serve the computational demand from time t to time $t + \tau - 1$ (the instance expires at time $t + \tau$). Just to give a real-life example, reserved instances are bought for six months period. Hence, $\tau = 24 \times 30 \times 6 = 4320$ ($24 \times 30 \times 6$ is the number of one hour in six months assuming that every month has 30 days). It is also important to point out that, in general:
$$p < P < p\tau$$
$p < P$ is obvious because reserved instances lasts for more time slots than on-demand instance and hence will cost more. $P < p\tau$ captures the idea of "bulk discount", i.e. rather than buying an on-demand instance consecutively for τ time slots (which costs $p\tau$), it is more cost effective to buy a reserved instance what will lasts τ time slots.

The agent MUST serve all its computational demand at every time slot. The objective of the agent is to strategically buy the instance types in each time slot in order to minimize the β-discounted cost where the cost in a time slot is the cost of buying reserved and on-demand instances in that time slot. Answer the following questions:

(a) Write an expression for A_t, the number of reserved instances bought **before** time t which did not expire at time t. **(2 marks)**

(b) What is the maximum number of either on-demand or reserved instance that is needed for this problem? Concentrate on the parameters D and M to answer this question. **(2 marks)**

(c) Using help from (a) and (b), and your understanding of the problem description, what is the state and state space for this problem? **(4 marks)**

(d) What is the action and action space for this problem? **(2 marks)**

(e) Define the cost for all state-action pair for this problem. **(3 marks)**

(f) What is the Bellman optimality equation for this problem? **(7 marks)**

Q3: Reinforcement Learning **(20 marks)**

Answer the following questions:

(a) Briefly explain Q-Learning and SARSA with suitable example? **(10 marks)**

(b) Explain the significance of Gamma, Alpha and Epsilon greedy in building reinforcement learning models? **(10 marks)**

Q4: Deep Reinforcement Learning **(20 marks)**

(a) During lecture, we saw the pseudocode for Deep Q-Learning. In this problem, you will write the pseudocode for **Deep SARSA** what uses DQN of **Architecture 1** (state and action as input and the Q-value of the state and action as output). More specifically, the skeleton of the pseudocode for Deep SARSA is given below. You have to fill lines 6, 8, 9, 10, 11, and 14. **You don't have to write the entire pseudocode in your answer script**. Just mention the line number and the pseudocode for that line in your answer script. **(8 marks)**

HINT: *(i)* Lines 6, 7, and 8 resembles SARSA of module 3. *(ii)* In line 10, the batch size is 1.

Algorithm 1: Psuedocode for Deep SARSA with DQN of Architecture 1

1 Initialize a predict DQN $\hat{Q}(\cdot;\phi_P)$ and target DQN $\hat{Q}(\cdot;\phi_T)$. Both the DQN should have the same architecture just different parameters.
2 Set an integer N_u (predict DQN update frequency), N_T (target DQN update frequency), N_b (training batch size), and $counter = 0$.
3 **for** *every episode until convergence* **do**
4 Reset the environment to get the current state x.
5 Choose learning rate, α, and exploration probability, ε, for this episode.
6 *Pick action: FILL IN THE BLANK.*
7 **for** *every time slot of the episode until convergence* **do**
8 *Take action: FILL IN THE BLANK.*
9 *Pick action: FILL IN THE BLANK.*
10 *Training data generation for Architecture 1: FILL IN THE BLANK.*
11 *Training DQN: FILL IN THE BLANK.*
12 **if** $counter\%N_T == 0$ **then**
13 Set $\phi_T \leftarrow \phi_P$.
14 *Update variables for the next time slot: FILL IN THE BLANK.*

(b) Mention three drawbacks of Deep SARSA over Deep Q-Learning. **(3 marks)**

HINT: Think about the fundamental difference between Deep Q-Learning and Deep SARSA. Then, think about how that difference helps in mitigating the issues that are there in "online" Deep Q-Learning.

(c) Select ALL the options that are true for Deep Q-Learning. Also, justify your answer:

(2 marks)

Option 1: Can deal with discrete state space and discrete action space.
Option 2: Can deal with discrete state space and continuous action space.
Option 3: Can deal with continuous state space and discrete action space.
Option 4: Can deal with continuous state space and continuous action space.

The remaining questions of Q4 is about Actor-Critic algorithm. In Actor-Critic algorithm, there is an actor (neural) network and a critic (neural) network. The hyperparameters of the actor and critic network are as follows:

Actor network: Input layers has a size of 30. There are two hidden layers, each of 50 neurons. The output layer has 16 neurons.

Critic network: There is only one hidden layer containing 50 neurons.

(d) What is the activation function in the last layer of critic network? Justify your answer.

(2 marks)

(e) What kind of loss function will you use to train the actor network? Justify your answer.

(2 marks)

(f) What is the total number of parameters (weights and biases) in the critic network? **(3 marks)**

Q5: State Estimation **(20 marks)**

(a) Consider a Partially Observable Markov Decision setup where one wants to get the probability of the current state given the history of observations and actions, as well as the current observation. Suggest an approach to do so with necessary derivations. **(13 marks)**

(b) Consider a system whose dynamics and observation are governed by the following equations:

$$x_t = 0.8x_{t-1} + 2u_{t-1} + \varepsilon_{t-1}$$
$$z_t = x_t + \delta_t$$

In the above two equations, all the variables are **real numbers** and **scalar**. x_t, z_t, and u_t are the state, observation, and control input respectively. ε_t and δ_t are Gaussian noise. The mean and standard deviation of ε_t are $\mathbf{0.5}$ and 4 respectively. The mean and standard deviation of δ_t are 0 and 2 respectively. Using **Kalman filter**, derive a recursive update rule to estimate state x. A good estimate of a Gaussian random variable is its mean as it has the highest probability. **Caution:** Kalman filter can't **directly** deal with noises that has non-zero mean.