Name: Sameena Simmin RollNo: 21481A5486 class: IV AI&DS (B-section).

Reinforcement Learning

1) How does Reinforcement learning different from supervised and
a) unsupervised learning? write the key features of Reinforcement learning.

Ans: Supervised Learning is the most basic and widely used type of Machine learning. A model is trained on a dataset where the correct output or "label" is already provided for each input.

→ For example, imagine you have a dataset of pictures of cats and dogs. The labels for the dataset would be "cat" and "dog". The model is then able to use this information to make predictions about new pictures of cats and dogs it has never seen before.

→ Most well-known and commonly used algorithms include:

1. Linear Regression
2. Logistic Regression
3. Decision Trees
4. Random Forest
5. Support Vector Machines (SVM)
6. K-Nearest Neighbors (KNN)

Unsupervised learning is when the model is given a dataset without any labels or output. The model must then find patterns and structure, within the data on it's own.

→ Common example is clustering, where a model groups similar data points together. Imagine you have a dataset of customer data. The model would group customers based on similar characteristics such as age, location, and spending habits.

→ popular learning algorithms are:

1. K-means
2. Hierarchical Clustering
3. PCA (Principal component Analysis)

4. t-SNE (t-Distributed Stochastic Neighbor Embedding).

Reinforcement learning is a bit different from supervised and unsupervised learning. The model learns from the consequences of its actions. The model receives feedback on its performance, and uses that information to adjust its actions and improve its performance over time.

→ Example is training a model to play a game like chess or Go. The model receives feedback on its performance in the form of win/loss, and then adjusts its strategy to improve its chances of winning.

→ RL algorithms are:

1. Q-learning
2. SARSA
3. DQN
4. A3C.

Key features of Reinforcement learning:

- Exploration vs Exploitation: Balancing b/w trying new actions and exploiting known strategies.
- Delayed Rewards and Temporal Difference learning: Handling situations where rewards are not immediate.
- Model-Based vs. Model-Free learning: Approaches based on whether the agent builds a model of the environment.
- Trial-and-Error learning: Learning from experience rather than predefined datasets.

(1) b) Briefly explain the basic elements of Reinforcement learning.

Ans: The elements of Reinforcement learning are divided into four types:

i, Policy: Defines the agent's behavior at a given time.

ii, Reward function: Defines the goal of the RL problem by providing feedback

iii, Value function: Estimates long-term rewards from a state

iv, Model of the Environment: Helps in predicting future states and rewards for planning

② a) write the value function used by the RL agent to compute the value of a state in tic-tac-toe game. Show how this value function works with a suitable example.

Ans: In Reinforcement learning, the value function is used to estimate the value of a particular state in a game like Tic-Tac-Toe. The value of a state represents how good it is for the agent to be in that state, based on the expected future rewards.

→ For Tic-Tac-Toe, the value function V(S) of a state 's' can be defined as follows:

- V(S) = 1 ; if state 's' is winning state for the agent
- V(S) = 0 ; if state 's' is a draw.
- V(S) = -1 ; if the state 's' is a losing state for the agent.
- For non-terminal states, the value function can be computed based on the expected outcome of the game from that state.

Example:

1. Terminal states:

- winning state for the agent (X):

x | x | x → player 'x' has won, so: V(S) = 1

O | | O

  | O |

- Losing state for the agent (x):

```
O | O | O
X | X |
  |   | X
```

→ player 'o' has won, so: $V(S) = -1$

- Draw state:

```
X | O | X
X | O | O
O | X | X
```

→ It's a draw, $V(s) = 0$

2. Non-terminal States:

```
X | O | X
  | X | O
  | O |
```

→ The value of this state $V(s)$ would depend on the values of possible next states:

- If 'x' plays in the bottom-right corner and wins, the value of that state is 1.
- If 'x' plays in the center-left, and the opponent plays optimally, the game might end in a draw (value 0)
- If 'x' makes a mistake and the opponent wins, the value will be -1.

$$\boxed{V(s) = \max_{a} \sum_{s'} P(s'|s,a) \cdot V(s')}$$

where, s' = possible future state after taking action 'a'.

$P(s'|s,a)$ = probability of transitioning the state s' from state 's' after action 'a'

$V(s')$ = value of the future state s'.

Thus, the agent will choose the action that leads to the highest possible value.

---

(2) b) What is an n-arm bandit problem? Describe any one solution to solve the problem.

Ans: The n-armed bandit problem is a classical problem in RL and decision theory. It is based on the metaphor of a gambler facing 'n' slot machines, each with a different and unknown probability distribution of rewards.

→ The goal is to maximize the total reward by playing the machines over time, balancing exploration and exploitation.

Problem SetUp:

- You have 'n' slot machines, each with an unknown probability destribution of reward
- Every time you pull the lever of a machine, you receive a reward based on the probability distribution of that machine.
- Our objective is to maximize the total reward over multiple plays by selecting which machine to pull at each step

Solution: The ϵ-greedy Algorithm:

One popular solution to the n-armed bandit problem is the ϵ-greedy algorithm. This algorithm introduces a balance b/w exploration and exploitation in a simple way.

1. Initialization - start by assigning an initial estimate of the expected reward for each machine.
2. play a Machine - explore (probability ϵ), exploit (probability 1-ϵ)
3. Update the estimates:

$$Q_{K+1}(a) = Q_K(a) + \frac{1}{N(a)}(R(a) - Q_K(a))$$

→ The n-armed bandit problem is a fundamental example of the exploration Vs. exploitation trade-off in RL.

→ The ϵ-greedy algorithm provides a simple yet effective solution, balancing the need to explore different actions with the desire to exploit the best-known action to maximize the rewards over time

3) Describe gradient bandit algorithm. Derive an equation to update the preference value of an action using stochastic approximation.

Ans: The gradient bandit algorithm is a method for solving the multi-armed bandit problem by using preferences to guide the selection of actions.

→ Instead of directly estimating the value of each action, the algorithm assigns a preference to each action, which is adjusted using gradient ascent to maximize expected rewards

→ Steps in Gradient Bandit Algorithm:

1. Initialize preferences $H(a)$
2. Compute the action probabilities using softmax function:

$$\boxed{\pi(a) = \frac{e^{H(a)}}{\sum_{b=1}^{n} e^{H(b)}}}$$

3. Select an action based on $\pi(a)$
4. Receive a reward 'R'
5. Update the preferences.

Deriving the Update Equation:

The gradient of the expected reward $E[R]$ with respect to the preferences $H(a)$ can be written as:

$$\frac{\partial E[R]}{\partial H(a)} = E\left[(R - \bar{R}) \frac{\partial \pi(a)}{\partial H(a)}\right]$$

→ using softmax function for the probabilities $\pi(a)$, we can derive the gradient:

$$\frac{\partial \pi(a)}{\partial H(a)} = \pi(a)(1 - \pi(a))$$

$$\frac{\partial \pi(b)}{\partial H(a)} = -\pi(a)\pi(b)$$

Hence the gradient bandit algorithm:

- uses preferences for each action instead of estimating values.
- selects actions using the softmax function based on preferences
- updates preferences using gradient ascent based on the rewards received and the probability of selecting the action.
- Is useful for optimizing decisions in uncertain environment like the multi-armed bandit problem.

---

(4) What do you mean by a discounted return? why it is important?
a) write the equation for computing the discounted return.

Ans: A discounted return is the sum of future rewards in a reinforcement learning task, where each future reward is multiplied by a discount factor 'γ' (b/w '0' and '1').

→ The discount factor reduces the weight of future rewards, so that rewards received sooner are valued more than rewards received later.

→ why it's Important:

- Immediate rewards are often more important than distant future rewards in decision-making.
- The discount factor 'γ' helps to model this by reducing the importance of future rewards.
- It also ensures that the sum of infinite future rewards converges to a finite value, preventing unstable calculations.

Equation for Discounted Return:

If $R_1, R_2, R_3 \ldots$ are the rewards at time steps 1, 2, 3 --

the discounted return G_t at time 't' is:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$$

→ In a more compact form, the equation is :

$$\boxed{G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}.}$$

where, γ = discount factor ($0 \leq \gamma \leq 1$)
R_t = reward at time step 't'.

→ In short, the discounted return helps balance short-term vs long-term rewards in reinforcement learning.

4 b) what is Markov property? Consider the following weather data.

		Tomorrow's Weather		
Today's weather		Sunny	Rainy	Cloudy
	Sunny	0.8	0.05	0.15
	Rainy	0.2	0.6	0.2
	cloudy	0.2	0.3	0.5

using Markov property, compute the following -

i, Given that today is cloudy, what is the probability that it will be rainy two days from now?

ii, Given that today is Rainy and yesterday was cloudy then what is the probability that tomorrow is sunny?

Ans: Markov Property:

The Markov property means that the future state only depends on the present state, not on past states.

i, Probability it will be rainy two days from now if today is cloudy:

- from cloudy to Rainy tomorrow:

  $P(\text{Rainy} / \text{cloudy}) = 0.3$

- from Rainy to Rainy the next day:

  $P(\text{Rainy} | \text{Rainy}) = 0.6$.

Total probability $= 0.3 \times 0.6 = 0.18$

ii, Probability that tommorow is Sunny, given today is Rainy and Yesterday was cloudy:

By the Markov property, only today's state matters. from Rainy today to sunny tomorrow:

$$P(Sunny / Rainy) = 0.2$$

Thus, the probability is 0.2 //.