Birla Institute of Technology & Science, Pilani
Work Integrated Learning Programmes Division
First Semester 2024-2025

Comprehensive Exam
(EC-3 Regular)

Q1) [**2+5 = 7 Marks**]

(a) Consider a reinforcement learning scenario: A self-driving car learns to navigate a city using reinforcement learning. The vehicle receives a reward of +100 for reaching its destination, -10 for stopping at a red light, and -1 for every second it spends driving. The agent's objective is to maximize the discounted cumulative reward.

Given this scenario, explain the role of the discount factor briefly [1.0 M]. If the $\boldsymbol{\gamma}$ is increased from 0.9 to 0.99, how would this change the self-driving car's decision-making in this scenario? Justify your answer. [1.0 M].

Answer:

The **discount factor (γ)** determines how much the agent values future rewards relative to immediate rewards. A higher γ places more emphasis on long-term rewards, while a lower γ prioritizes short-term gains. [1 Mark]

If **$\boldsymbol{\gamma}$ increases from 0.9 to 0.99**, the self-driving car will focus more on reaching the destination (+100) rather than minimizing immediate penalties (-1 per second, -10 for stopping). [1 Mark]

Additional note to TA on Marking Scheme:

Full marks can be given if the significance of the discount factor is explained and the difference for different discount factors is mentioned. Deduct with an appropriate comment if the answer is partially correct. Make a clear comment/feedback.

(b) A gambler plays a 4-armed bandit (slot machine with 4 levers) using the ϵ-greedy strategy with ϵ=0.2. The estimated values (Q-values) of the four arms at a given time step are:

$$Q_1 = 3.5, \quad Q_2 = 5.0, \quad Q_3 = 4.2, \quad Q_4 = 6.1$$

(b-1) What is the probability of selecting the optimal arm? [1.0 M].
(b-2) What is the probability of choosing each of the other arms? [1.0 M].
(b-3) Suppose the gambler chooses arm 2 and receives a reward of 7.0. Using the incremental update rule with a step size of α=0.1, calculate the updated Q_2 value. [1.0 M].
(b-4) Compute the ***expected reward at this step***, assuming the agent follows the ϵ-greedy policy and the true expected rewards of the arms are: $R_1 = 4.0$, $R_2 = 5.5$, $R_3 = 4.5$, $R_4 = 6.0$. [2.0 M].

Answer:

(b-1) The optimal arm is the one with the highest Q-value, which is Q4 = 6.1. In an ϵ-greedy strategy:

- The best action is chosen with probability $1 - \epsilon = 1 - 0.2 = 0.8$.
- Random selection among all 4 arms happens with probability $\epsilon = 0.2$, so each arm is chosen with 0.2 / 4 = 0.05 probability during exploration.
- Thus, the probability of selecting the optimal arm (Q4) is: 0.8 + 0.05 = **0.85**. [1 Mark]

(b-2) Probability of choosing each of the other arms

- Each non-optimal arm (Q1, Q2, Q3) is chosen only during exploration, with a probability of 0.05 each. Thus, the probability of choosing Q1, Q2, or Q3 is **0.05 each. [1 Mark]**

(b-3) Updated Q2 value

Using the incremental update rule: Qnew = Qold + α (R − Qold) => Q2 = 5.0 + 0.1 (7.0 − 5.0) = **5.2** [1 Mark]

(b-4) Expected reward at this step

The expected reward is calculated using the probabilities of selecting each arm and the corresponding true expected rewards:

E[R] = P(Q4) R4 + P(Q1) R1 + P(Q2) R2 + P(Q3) R3

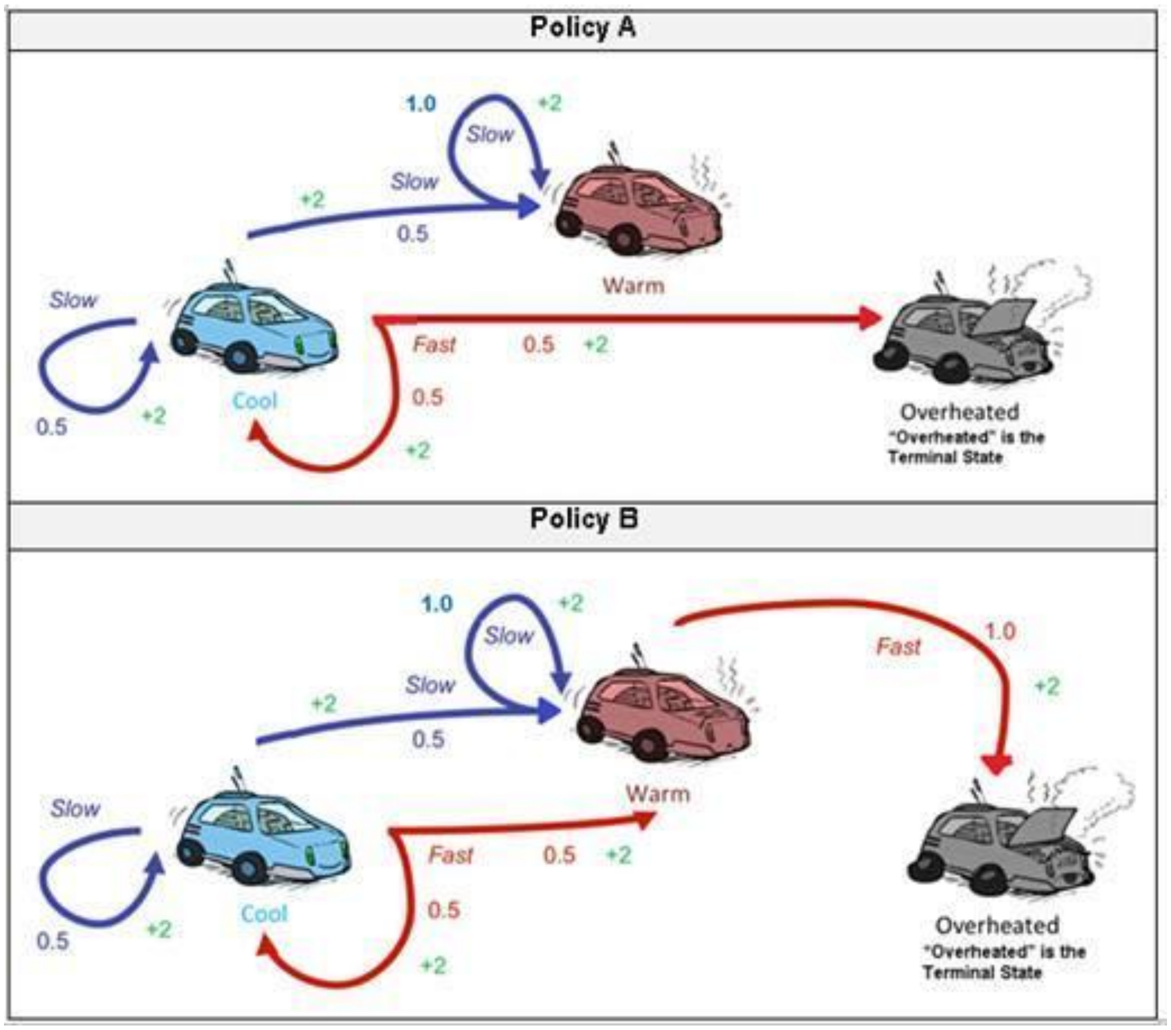
= (0.85×6.0) + (0.05×4.0) + (0.05×5.5) + (0.05×4.5) = **5.8 [2 Marks]**

Additional note to TA on Marking Scheme:

Give full marks if the steps and final answer is correct. Deduct with an appropriate comment if the answer is partially correct. Make a clear comment/feedback.

Q2) **[3+2+2= 7 Marks]**

(a) A racecar is learning to maximize the discounted cumulative reward while learning to drive effectively on a track. Two different policies (Policy A and Policy B) designed are shown below in state transition diagrams. Use γ = 1 and constant transition reward = +2.



Compare & comment on the suitability of the given policies A and B w.r.t dynamic programming. Justify your comment for each using no more than four well-defined statements. [3.0 M]

Answer:

Policy A is not the best design . It doesn't follow finite MDP and hence traditional DP algorithms like tabular approach may fail in this case.

Policy B is relatively good as it is guaranteed to transition to Terminal state from any other non-terminal state.

Additional note to TA on Marking Scheme:

implication of the no outgoing path from state "Warm" in Policy if identified by the students **1.5** marks can be awarded.

Finite MDP need if stated **1.5** mark can be awarded

Deduct with an appropriate comment if the answer is partially correct. Make a clear comment/feedback.

(b) Using policy A from above, apply asynchronous value iteration in this specific order = (V(Warm), V(Overheated), V(Cool)) for three iterations. Assume $V(S_0) = 6$ for all the states, including the terminal state. [2.0 M]

Answer:

For Given Order: Updates are expected as below:

Iteration	**V(Warm)**	**V(Overheated)**	**V(Cool)**	**Difference** **Delta(V(Warm)) = 2 always** **Delta(V(Overheated)) = 0 always** **Delta(V(Cool)) = *given below***
0th : Initial value	6	6	6	
1st	8	6	max(4+5 , 4+4) = 9	3
2nd	10	6	max(5.5+6 , 5.5+4) = 11.5	2.5
3rd	12	6	max(6.75+7 , 6.75+4) = 13.75	2.25 (4th Iteration it will be 2.125, 5th Iteration it will be ~1.0625)

Additional note to TA on Marking Scheme:

(1m) Atleast one bellman equation update for the V(Cool) must be clearly shown in detail is sufficient for marking . V(Cool) for three iterations must be calculated..

(1m) V(state) - For other states “Cool” & “Overheated”

Partial marks (1m in total) can be awarded for one or 2 iterations or 3 iterations with incorrect values if procedure is correct

(c) Observe the results of the three iterations and analyze if a thresholding of Δ <=2 is applied for all the states in the algorithm. Will the algorithm converge for the given set-up? Justify your answer in no more than two well-defined statements. **Note** : After iteration t, for every state “S”, $\Delta_t(S) = | V_{t-1}(S) - V_t(S) |$ is the difference between the value of the state between subsequent iterations [2.0 M]

Answer:

Check the above tabulated answer . The last column has the Delta values answer. Yes, with threshold it will converge : Delta for "Overheated" is always 0. Delta for "Warm" is always 2. Delta of "Cool" is a decreasing function. Hence it will converge in 4-5 iterations.

Additional note to TA on Marking Scheme:

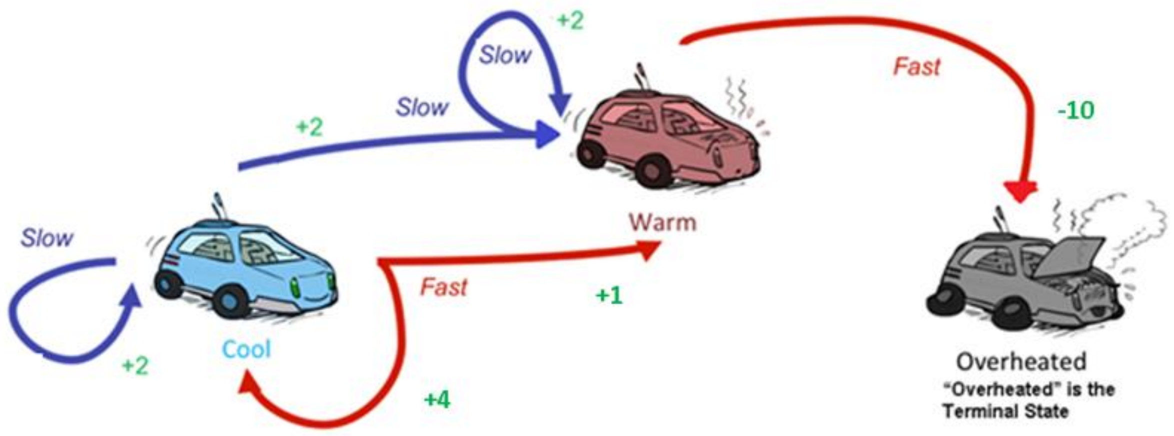
(1m) Delta value calculation must be correct. If the delta value for V(cool) is clearly shown with justification it is sufficient for marking.

(1m) Delta value calculation of other states must be correct.

Partial marks (1m in total) can be awarded for convergence identification but with incorrect/missing explicit delta value calculation.

Q3) [**4+1+2= 7 Marks**]

Consider the policy of an RL agent given below to answer the questions



(a) Apply on policy first visit Monte Carlo control find the value of the state "Cool" using below episode with γ = 0.5 and ϵ = 0.6. [4.0 M]

Episode : (Cool, Slow, Cool, Slow, Cool, Fast, Warm, Fast, Overheated)

Answer:

Init Values : Assume initial return as "0" as per the algorithm and must proceed

states	Cool		Warm		Overheated	Discount	0.5
Actions	Fast	Slow	Fast	Slow		Epsilon	0.6
Assume Old-value	0	0	0	0	0		
Reward	4 for next state "cool" 1 for next state "warm"	2	-10	2			

Loop for t=3 to t=0

@ t= 3, (warm, Fast, Overheated): Return(Warm , Fast) = Discount*G + Reward = (0.5*0) + (-10) = -10
@ t= 2, (Cool, Fast, Warm) :Return(Cool, Fast) = Discount*G + Reward = (0.5*-10) + (+1) = -4
@ t= 1, (Cool, Slow, Cool) : Return(Cool, Slow) = Discount*G + Reward = (0.5*-4) + (+2) = 0
@ t= 0, (Cool, Slow, Cool) : Return(Cool, Slow= Discount*G + Reward = (0.5*0) + (+2) = +2

Q(Cool, Fast) = -4
Q(Cool, Slow)= +2
V(Cool) = +2

Additional note to TA on Marking Scheme:

If the student has given any of the above V(cool) and/or Q(Cool, Slow) and/or Q(Cool, Fast), then it can be considered for full marking.

Each step in above = 1 mark

Since the first visit alone must be considered, students must not average repeat to Q(Cool, Slow) which is occurring twice.

(b) If the above problem is applied with an off-policy MC algorithm, justify the significance of weighted importance sampling compared to ordinary importance sampling. [1.0 M]

Answer:

In off-policy algo with ordinary importance sampling based episodes, whenever states are revisited in the trajectory or have loops , it suffers from infinite variance issues. This leads to a convergence issue. Whereas the weighted importance sampling algorithm produces a weighted average of only the returns consistent with the target policy, and all of these would be exactly 1. Hence weighted IS given stable convergence

Additional note to TA on Marking Scheme:

Identification of variance as a problem Ordinary IS is sufficient for marking.

(c) Explain a scenario where Temporal Difference learning becomes equivalent to Monte Carlo learning if applied to the given use case. [2.0 M]

Answer:

TD mimics MC if “n” in the n-step look ahead is set to infinity w.r.t time horizon.

Show any sample episode w.r.t to above car race state transition to illustrate working of n-step TD & relate to MC

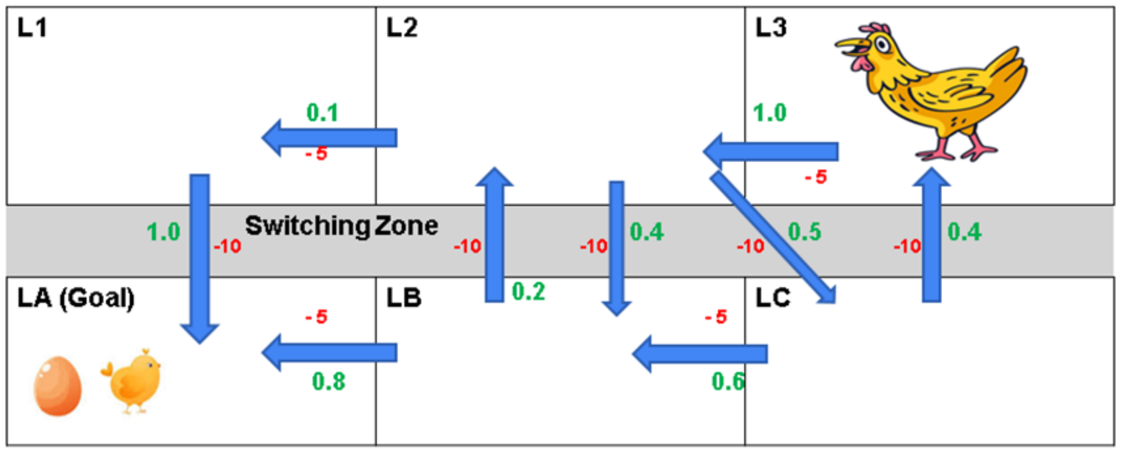
Additional note to TA on Marking Scheme:

(1m) For the n=infinity identification

(1m) Articulate the concept with relevancy only using above car race example

Q4) [5+2 = 7 Marks]

Consider the following grid-based environment where a chicken navigates through different locations (L1, L2, L3, LA, LB, and LC) to reach the goal state (LA). Movement is only permitted between states along the blue arrows. Each arrow (transition between states) is marked with its corresponding rewards (in red colour) and the probability of taking that action (in green colour). You can observe that the transitions leading to crossing the "Switching Zone" result in higher penalties. You can observe that the policy is implied by the probability of action (number written in green) for each state.



(a) Apply Q-Learning for one iteration with $\boldsymbol{\gamma}$ =1 and $\boldsymbol{\alpha}$ = 0.2. In an episode, use greedy action selection per time step. [5.0 M]

Answer:

Assume Q value as "0" and proceed!
$Q(s,a)=Q(s,a)+\alpha(R_{t+1}+\gamma\ \max_{a'}[Q(s',a')]-Q(s,a))$

@time step 1 : Agent moves from L3 to L2. (L3, west, L2, -5)
Q(L3,West) = 0+0.2(-5+1*max(0,0,0)-0) = -1

@time step 2 :
Max((-5*0.1, -10*0.5, -10*0.4) = -0.5 . Hence chooses to go to L1 : (L2, west, L1, -5)
Q(L2,West) = 0+0.2(-5+1*max(0)-0) = -1

@time step 3 : Agent moves from L1 to LA automatically as there is only one outgoing action . (L1, south, LA(Goal) , -10)
Q(L2,West) = 0+0.2(-10+1*max(0)-0) = -2

Path using greedy = L3-->L2-->L1-->LA

Additional note to TA on Marking Scheme:

(1 mark) Highlighted greedy action selection step from state L2 is mandatory
(1 mark each) For timesteps 1, 2 & 3
(1 mark) Clear formula application & correct Values

Important Note: If the students have assumed arbitrary Q - value then you have substitute to verify corresponding answers.

(b) Assume that the chicken uses the policy where the action with the greatest Q-value is always preferred. Is this an optimal policy? Why or Why not? Explain in no more than three well-formed sentences. [2.0 M]

Answer:

The policy with the greatest Q values is not optimal, because it suffers from maximization bias. When the next state actions with maximum values are selected, it leads to bias called maximization bias.

Additional note to TA on Marking Scheme:

maximization bias is the key word.

Q5) [4+ 3 = 7 Marks]

a) Consider the following scenarios:

Scenario 01: Tic-Tac-Toe game environment with a small, finite state space.

Scenario 02: Autonomous driving in a real-world city with a large, continuous state space.

Answer the following questions:

a-1. Which methods (Tabular or Function Approximation) are appropriate for the given scenarios? Justify your selection. [1.0 M]

a-2. Why are Semi-Gradient methods preferred over Stochastic Gradient methods for value function learning in continuous spaces? Provide justification. [1.0 M]

a-3. Linear function approximators often require handcrafted, domain-dependent features, which depend on expert knowledge. Why is this considered a limitation? How do deep learning-based function approximators address this issue through automatic feature extraction? Mention two techniques used in deep learning for automatic feature extraction. [2.0 M]

Answer:

a-1:

- Scenario 01 (Tic-Tac-Toe): Tabular methods are appropriate because the state space is small and finite, allowing explicit storage and updates of state-action values. (0.5M)
- Scenario 02 (Autonomous Driving): Function approximation is necessary due to the large and continuous state space, where storing values for all possible states is infeasible. Instead, function approximators like neural networks generalize across similar states. (0.5M)

Marking Scheme: Full marks if both scenarios are correctly justified. Partial marks if justification lacks clarity.

a-2

- The Semi-Gradient methods are preferred because they update value function approximations using bootstrapped estimates (TD learning), which enables efficient learning from sampled transitions without needing the true return. In contrast, Stochastic Gradient methods rely on supervised learning signals, which may not always be available in reinforcement learning settings. (0.5+0.5M)

Marking Scheme: Full marks for mentioning bootstrapping and efficiency in learning. Partial marks if reasoning is incomplete.

a-3

- Limitation of handcrafted features: Handcrafted features require expert domain knowledge, are task-specific, and may not generalize well to unseen scenarios, limiting scalability. (0.5M)
- Deep learning-based solution: Deep learning models, such as CNNs and Transformers, learn hierarchical and abstract representations automatically from raw data, eliminating the need for manual feature engineering. (0.5M)
- Techniques used:
  1. Convolutional Neural Networks (CNNs): Automatically extract spatial features from image-based inputs. (0.5M)
  2. Autoencoders: Learn compressed, meaningful representations of data in an unsupervised manner. (0.5M)

Marking Scheme: Full marks if limitations, deep learning advantages, and two correct techniques are mentioned. Partial marks for missing or incorrect techniques.

`

b) Answer the following

b-1. REINFORCE and REINFORCE with baseline are both policy gradient methods. What is the key difference between REINFORCE and REINFORCE with baseline, and how does the baseline help in reducing variance? [1.0 M]

Answer:

- Key Difference: REINFORCE directly uses the return Gt as the reward signal for updating the policy, while REINFORCE with baseline subtracts a baseline value b(s) (typically the state value function V(s) from Gt before computing the policy gradient. (0.5M)
- Variance Reduction: The baseline helps reduce variance by centering the advantage estimate, reducing fluctuations in gradient updates without introducing bias, leading to more stable learning. (0.5M)

Marking Scheme: Full marks if the difference is clearly stated and variance reduction is explained. Partial marks if variance reduction reasoning is incomplete.

b-2. Imagine a robot navigating a maze to reach a goal. [2.0 M]

a. What is the role of the actor and critic r in the Actor-Critic method for the robot navigation scenario?
b. Why is it typically impractical to run policy evaluation and policy improvement steps to convergence in large-scale reinforcement learning problems?

Answer:

b-2

a. Role of Actor and Critic in Actor-Critic Method:

- Actor: The actor represents the policy and decides the robot's next action based on the current state. It learns by adjusting the policy parameters using feedback from the critic. (0.5M)
- Critic: The critic evaluates the action taken by the actor by estimating the value function. It provides feedback in the form of the advantage estimate, guiding the actor to make better decisions. (0.5M)

b.

- Computational Infeasibility: Running policy evaluation to convergence in each iteration is computationally expensive, especially with large state-action spaces.
- Non-Stationarity: The policy is continuously evolving, making it impractical to wait for convergence at each step before updating the policy.
- Exploration-Exploitation Tradeoff: In complex environments, extensive evaluation delays policy updates, potentially hindering learning progress.

Marking Scheme: Full marks if both the roles of actor and critic and the reasons for impracticality of full convergence are correctly explained. Partial marks if one part is missing or unclear.

Q6) [3+ 2 = 5 Marks]

(a) How AlphaGoZero with a single network outperforms AlphaGo with dedicated network for policy & value? Explain the role of MCTS in both the RL programs. [3.0 M]

Answer:

AlphaGo Zero outperforms AlphaGo despite using a **single neural network** instead of separate **policy and value networks**. The key reasons for this improvement are:

1. **Unified Representation**
2. **Self-Play with Reinforcement Learning**
3. **Better Neural Network Architecture**
4. **More Efficient Monte Carlo Tree Search (MCTS)**
5. **Greater Training Stability**

Role of MCTS in Both RL Programs

AlphaGo

- MCTS searches through possible moves and **relies on a policy network** to guide simulations.
- The **value network evaluates board states**, helping MCTS prune unpromising paths.
- Because policy and value are computed separately, it requires **more computation and training data**.

AlphaGo Zero

- MCTS is **directly informed by a single network**, which **outputs both move probabilities and value estimates**.
- This **reduces computation time** and enables **more efficient searches**.
- MCTS performs **more simulations per second** and improves move selection.

Additional note to TA on Marking Scheme:

AlphaGo Zero surpasses AlphaGo by **removing human biases, using a single efficient network, leveraging self-play learning, and improving MCTS efficiency**. The **integration of policy and value estimation** into a **single network** leads to **faster, more effective decision-making** and ultimately **stronger gameplay**.

(b) Explain why DAGGER is a reinforcement learning algorithm with two well-articulated points [2.0 M]

Answer:

1. DAGGER Uses an Interactive Learning Process with an Expert (Sequential Decision Making)

2. DAGGER Aims to Optimize Long-Term Decision Outcomes (Like RL)

Thus, while DAGGER is technically a form of **imitation learning**, its **interactive nature** and **focus on long-term sequential decision-making** make it closely related to reinforcement learning.

Additional note to TA on Marking Scheme:

DAGGER is considered an RL algorithm because it **interactively learns from an expert in a sequential decision-making process** rather than relying on a static dataset. Additionally, it **optimizes long-term decision outcomes by refining the policy iteratively**, preventing error accumulation over multiple steps.