ABHYASMITRA.IN
COMPREHENSIVE UNIVERSITY STUDY NOTES

MACHINE LEARNING

Unit V — Reinforcement Learning

Topics Covered in this Unit

Introduction to Reinforcement Learning (RL), Need for RL & Core Architecture Components
Comprehensive Comparison: Supervised vs. Unsupervised vs. Reinforcement Learning
Applications & Grand Challenges: Delayed Rewards, Credit Assignment & Exploration Dilemma
Markov Decision Process (MDP): The Markov Property, 5-Tuple Formulation & Discount Factor gamma
Task Environments: Episodic Tasks (Terminal States) vs. Continuing Tasks (Infinite Horizons)
RL Mathematical Framework: Policies (pi), State-Value V(s) & Action-Value Q(s, a) Functions
Bellman Expectation Equations & Bellman Optimality Equations for V*(s) and Q*(s, a)
Exploration vs. Exploitation Tradeoff & The Epsilon-Greedy Action Selection Strategy
Dynamic Programming (Model-Based) vs. Model-Free Temporal Difference Learning
The Q-Learning Algorithm: Off-Policy TD Control, Q-Table Mechanics & Bellman Update Rule
Case Study & Game Playing: Reinforcement Learning Implementation for Tic-Tac-Toe
High-Yield University Exam Question Bank with Derivations, Equations & Unit Summary

Compiled in a structured, easy-to-understand, textbook style format
Specially curated for B.Tech / B.E. / BCA / MCA / B.Sc Computer Science & Data Science Students

COURSE SYLLABUS & MAPPING

[UNIT V SYLLABUS — PRESCRIBED UNIVERSITY CURRICULUM]

This section contains the official syllabus for Unit V. Verify with your university curriculum mapping.

Unit V Syllabus Breakdown:

Unit V - Reinforcement Learning: Introduction, need of reinforcement learning, components of reinforcement learning, comparison with supervised and unsupervised learning, applications and challenges of reinforcement learning. Markov Decision Process: Markov property, elements of MDP, episodic and continuing tasks. Reinforcement Learning Framework: Policy, state value function, action value function, Bellman equation, optimal policy. Reinforcement Learning Algorithms: Exploration vs Exploitation, ε-greedy strategy, dynamic programming, Q-Learning algorithm and update rule, simple reinforcement learning for game playing- Tic-Tac-Toe.

• Course Code: _____ • Semester / Year: _____

TABLE OF CONTENTS

5.1 Introduction to Reinforcement Learning

Definition: Reinforcement Learning (Sutton & Barto)

Reinforcement Learning (RL) is a goal-directed computational learning paradigm where an autonomous decision-making entity (the Agent) learns how to map situations to actions so as to maximize a numerical Reward signal through active trial-and-error interaction with a dynamic, uncertain Environment.

The Reinforcement Learning Architecture & Components

THE REINFORCEMENT LEARNING FEEDBACK LOOP

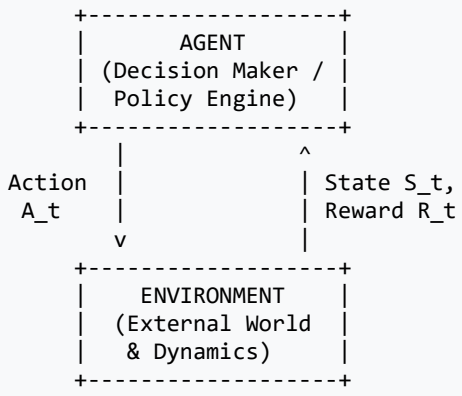

Figure: The Agent-Environment interaction loop across discrete time steps t = 0, 1, 2, ...

- **1. Agent:** The autonomous learner and decision-maker (e.g., a chess-playing algorithm, self-driving controller, robotic arm).
- **2. Environment:** Everything external to the agent with which the agent interacts (e.g., the physical road, chess board, stock market).
- **3. State (S_t in S):** A complete mathematical representation of the environment's condition at time step t.
- **4. Action (A_t in A(S_t)):** The set of all permissible moves or decisions available to the agent in state S_t.
- **5. Reward (R_{t+1} in R):** A scalar numerical feedback signal returned by the environment indicating the immediate desirability of transition S_t -> S_{t+1} under action A_t.
- **6. Policy (pi(a|s)):** The agent's internal behavior strategy; a mapping from perceived states to actions.
- **7. Value Function (V(s) & Q(s, a)):** Estimates the expected long-term cumulative future reward from a state, balancing immediate vs delayed returns.

- **8. Model of Environment:** Optional component that mimics environment behavior, predicting next states P(s'|s, a) and rewards R(s, a).

Master Comparison: Supervised vs. Unsupervised vs. Reinforcement Learning

Comparison Dimension	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Data Modality	Static dataset with ground-truth input-output pairs (X, y)	Static dataset with unlabelled feature vectors (X)	No prior dataset; sequential active environmental interaction
Supervision Signal	Explicit instantaneous target labels from human expert	None; self-organizing pattern and density discovery	Evaluative delayed scalar reward/penalty feedback
Temporal Sequence	I.I.D. assumption (Independent & Identically Distributed)	I.I.D. assumption across data instances	Sequential non-I.I.D. time series (Current action affects future states!)
Decision Impact	Predictions have zero impact on future data inputs	Clustering has zero impact on incoming data	Agent actions actively alter the future state of the environment
Core Goal	Minimize predictive Loss (MSE / Cross-Entropy)	Maximize data likelihood or minimize WCSS	Maximize expected cumulative discounted return G_t

Applications and Grand Challenges of Reinforcement Learning

- **Major Applications:** Game Playing (AlphaGo, Chess, Atari), Autonomous Drone Flight & Robotics Control, Algorithmic High-Frequency Trading, Smart Traffic Grid Optimization, Datacenter HVAC Cooling Energy Optimization (DeepMind).
- **Grand Challenges in RL:**
  - **• Delayed Reward & Credit Assignment Problem:** An action taken at step 5 may lead to a win at step 50. Determining which specific past action was responsible for the ultimate reward is mathematically challenging.
  - **• Exploration vs. Exploitation Dilemma:** Balancing the exploitation of known profitable actions against exploring unknown actions that might yield higher long-term gains.
  - **• High Sample Complexity:** Requires millions of trial interactions to converge; risky in real-world physical systems (e.g., self-driving cars cannot crash to learn).

5.2 Markov Decision Process (MDP)

Definition: The Markov Property

A state transition process is said to possess the Markov Property (Memoryless) if the future state depends ONLY upon the current state and action, and is conditionally independent of all past historical states and actions:

P(S_{t+1} = s', R_{t+1} = r | S_t, A_t, S_{t-1}, A_{t-1}, ..., S_0, A_0) = P(S_{t+1} = s', R_{t+1} = r | S_t, A_t).

The Formal 5-Tuple Formulation of an MDP

A Markov Decision Process is formally defined as a mathematical 5-tuple: (S, A, P, R, gamma):

- **1. S (State Space):** A finite set of all valid environmental states.
- **2. A (Action Space):** A finite set of all valid actions available to the agent.
- **3. P (State Transition Probability Matrix):** P_{ss'}^a = P(S_{t+1} = s' | S_t = s, A_t = a) (the probability of reaching state s' when executing action a in state s).
- **4. R (Reward Function):** R_s^a = E[R_{t+1} | S_t = s, A_t = a] (the expected immediate reward).
- **5. gamma (Discount Factor, gamma in [0, 1]):** Determines the present value of future rewards. When gamma = 0, the agent is 'myopic' (cares only about immediate reward R_{t+1}); as gamma -> 1, the agent becomes 'far-sighted', prioritizing long-term cumulative returns. Ensures mathematical convergence of infinite series.

Episodic vs. Continuing Tasks

- **1. Episodic Tasks:** Naturally break into distinct, finite episodes that always end in an absorbing Terminal State T (e.g., Checkmate in Chess, Game Over in Pacman). Total Return: G_t = sum_{k=0}^{T - t - 1} gamma^k * R_{t+k+1}.
- **2. Continuing Tasks:** Ongoing interactions that proceed indefinitely without termination (e.g., continuous robotic locomotion, chemical factory process temperature control). Total Return: G_t = sum_{k=0}^infinity gamma^k * R_{t+k+1} (discount factor gamma < 1 guarantees G_t is finite).

5.3 RL Framework & The Bellman Equations

1. Policies and Value Functions

- **Policy (pi):**
  - Deterministic Policy: a = pi(s) (maps state s directly to a single action a).
  - Stochastic Policy: pi(a|s) = P(A_t = a | S_t = s) (probability distribution over actions).

- **State-Value Function V^pi(s):** The expected total return starting from state s and following policy pi thereafter: V^pi(s) = E_pi [G_t | S_t = s].
- **Action-Value Function (Q-Function) Q^pi(s, a):** The expected return starting from state s, taking arbitrary action a, and thereafter following policy pi: Q^pi(s, a) = E_pi [G_t | S_t = s, A_t = a].

2. The Bellman Expectation Equations

The fundamental insight of Richard Bellman is that value functions can be decomposed recursively into the Immediate Reward plus the Discounted Value of the Successor State:

Bellman Expectation Equation for State Value V^pi(s)

*V^pi(s) = sum_{a in A} pi(a|s) * sum_{s', r} P(s', r | s, a) * [r + gamma * V^pi(s')]*

Meaning: The value of state s is the expected immediate reward r plus the discounted value of the next state V(s'), averaged over all policy actions.

Bellman Optimality Equations (For Optimal Policy pi*)

V(s) = max_{a in A} sum_{s', r} P(s', r | s, a) * [r + gamma * V*(s')]*

Q(s, a) = sum_{s', r} P(s', r | s, a) * [r + gamma * max_{a'} Q*(s', a')]*

Optimal Policy Extraction: pi(s) = argmax_{a in A} Q*(s, a).*

5.4 Reinforcement Learning Algorithms: Q-Learning

1. Exploration vs. Exploitation Tradeoff & Epsilon-Greedy Strategy

An agent must balance Exploitation (choosing the action with highest known Q-value to maximize reward) and Exploration (choosing random/novel actions to gather new information about the environment).

- **The Epsilon-Greedy Action Selection Strategy:**
  - With probability (1 - epsilon): Select Greedy Action a = argmax_a Q(s, a) (Exploitation).
  - With probability epsilon: Select a purely random action from Action Space A (Exploration).
  - Epsilon Decay: epsilon is initialized high (e.g., epsilon = 1.0) for pure early exploration, and gradually decayed over training episodes toward 0.01 for mature exploitation.

2. Dynamic Programming (Model-Based)

Dynamic Programming algorithms (Policy Iteration and Value Iteration) compute exact optimal value functions V*(s) by turning Bellman equations into iterative update assignments. Limitation: Requires a complete, perfect mathematical model of transition probabilities P(s'|s, a) and rewards R.

3. The Q-Learning Algorithm (Watkins, 1989 — Model-Free TD Control)

Q-Learning is an Off-Policy, Model-Free Temporal Difference (TD) algorithm that learns the optimal action-value function Q*(s, a) directly from environment transitions WITHOUT knowing transition probabilities.

The Q-Learning Bellman Update Rule (Classic Exam Formula)

*Q(S_t, A_t) := Q(S_t, A_t) + alpha * [R_{t+1} + gamma * max_a Q(S_{t+1}, a) - Q(S_t, A_t)]*

- *alpha in (0, 1]: Learning Rate (controls speed of value adaptation).*
- *gamma in [0, 1]: Discount Factor (weights future rewards).*
- *R_{t+1} + gamma * max_a Q(S_{t+1}, a): The TD Target (Target estimate of future return).*
- *delta_t = [TD Target - Q(S_t, A_t)]: The Temporal Difference (TD) Error.*

Step-by-Step Q-Learning Algorithm:

- **1. Initialization:** Initialize Q(s, a) table arbitrarily (e.g., all zeros) for all s in S, a in A, and Q(terminal, .) = 0.
- **2. For each episode:** Initialize start state S.
- **3. For each step of episode:**
  - Choose action A from state S using epsilon-greedy policy derived from current Q-table.
  - Take action A, observe immediate reward R and successor state S'.
  - Update Q-value: Q(S, A) := Q(S, A) + alpha * [R + gamma * max_a Q(S', a) - Q(S, A)].
  - Set S := S'.
  - Repeat until S is terminal state.

5.5 Case Study: Reinforcement Learning in Tic-Tac-Toe

Tic-Tac-Toe serves as the canonical foundational case study for understanding tabular reinforcement learning and value iteration.

- **1. State Space Representation:** The board is a 3x3 grid (9 squares, each containing 'X', 'O', or Blank). The total theoretical state space is 3^9 = 19,683 states (reduced to ~5,478 reachable legal board configurations considering symmetry and game termination).

- **2. Value Function Initialization:** The agent maintains a Value Table V(s) estimating the probability of winning from board state s:
  - V(s) = 1.0 (for terminal states representing a WIN for the agent).
  - V(s) = 0.0 (for terminal states representing a LOSS for the agent).
  - V(s) = 0.5 (for terminal DRAW states and all intermediate non-terminal states).
- **3. Gameplay & TD Learning Rule:** The agent plays thousands of training games against an opponent (or self-play). Most moves are greedy (argmax V(s')), with occasional exploratory random moves (epsilon).
- **Temporal Difference Value Update:** After making a move from state s to state s', the agent updates the value of the earlier state based on the current state: V(s) := V(s) + alpha * [V(s') - V(s)]. Through repeated games, win/loss feedback propagates backward from terminal states to opening moves!

5.6 High-Yield University Exam Question Bank

Part A: Short Answer Questions (2 to 5 Marks)

Q1. What is the Markov Property? [2-3 Marks]

- **Answer:** A state possesses the Markov Property if future transitions and rewards depend strictly on the current state S_t and action A_t, and are conditionally independent of all past historical states and actions: P(S_{t+1}|S_t, A_t, ..., S_0, A_0) = P(S_{t+1}|S_t, A_t).

Q2. Explain the Exploration vs. Exploitation dilemma and the epsilon-greedy strategy. [4 Marks]

- **Answer:** Exploitation maximizes immediate reward by picking the best known action (argmax Q(s,a)); Exploration tries novel actions to find better long-term strategies. Epsilon-greedy chooses the greedy action with probability (1 - epsilon) and a random exploratory action with probability epsilon.

Q3. State the Q-Learning update rule and define all terms. [4 Marks]

- **Answer:** Q(S_t, A_t) := Q(S_t, A_t) + alpha * [R_{t+1} + gamma * max_a Q(S_{t+1}, a) - Q(S_t, A_t)], where alpha is learning rate, gamma is discount factor, R_{t+1} is immediate reward, and the bracketed expression is the Temporal Difference (TD) Error.

Part B: Long Answer Questions (10 to 15 Marks Outline Guides)

- **Q4. Explain the Reinforcement Learning framework with Agent-Environment feedback loop, State, Action, Reward, Policy, and Value Function. Compare RL with Supervised and Unsupervised Learning. [12 Marks]**
- **Q5. Formulate a Markov Decision Process (MDP) as a 5-tuple (S, A, P, R, gamma). Differentiate between Episodic and Continuing tasks. [10 Marks]**
- **Q6. Derive the Bellman Expectation Equation for State Value V(s) and Bellman Optimality Equations for V*(s) and Q*(s, a). [12 Marks]**
- **Q7. Explain the complete Q-Learning algorithm step-by-step. Discuss how Reinforcement Learning is applied to solve Tic-Tac-Toe game playing with TD value updates. [12 Marks]**

5.7 Unit Summary & Key Takeaways

- Reinforcement learning trains autonomous agents to maximize cumulative scalar reward through environmental trial-and-error.
- An MDP is defined by 5-tuple (S, A, P, R, gamma), relying on the memoryless Markov Property.
- Discount factor gamma in [0, 1] balances immediate vs future rewards and guarantees finite returns in continuing tasks.
- Value functions V(s) and Q(s, a) satisfy Bellman Expectation and Optimality Equations.
- Epsilon-greedy balances exploration (probability epsilon) and exploitation (probability 1 - epsilon).
- Q-Learning is a model-free off-policy TD algorithm updating Q(S,A) via TD Error [R + gamma * max Q(S',a) - Q(S,A)].
- Tic-Tac-Toe utilizes tabular states and TD value updates V(s) := V(s) + alpha[V(s') - V(s)] to learn winning strategies.