Module 24: Reinforcement Learning

Learning outcomes

1. Identify Analyse reinforcement learning principles and their connections to biological learning systems and decision-making processes.
2. Implement MAB solutions in Python using different algorithms.
3. Apply MDPs and dynamic programming techniques to solve sequential decision-making problems.
4. Evaluate Q-learning algorithms and their applications in reinforcement learning scenarios.
5. Analyse optimisation strategies using reinforcement learning techniques for real-world applications and hyperparameter tuning challenges.

RL overview

- RL is a method by which an agent learns optimal actions through interaction with an environment.
- At each step, the agent observes the current **state**, selects an **action**, receives a **reward** and updates its strategy to maximise cumulative future reward.
- Unlike supervised learning, RL does not rely on labelled data – learning occurs through **trial and error** guided by feedback signals.
- The key challenge in RL is the **exploration–exploitation trade-off:** exploring new actions to discover better outcomes versus exploiting current knowledge to maximise immediate rewards.

The multi-arm bandit (MAB) problem

- The MAB problem captures single-step decision-making under uncertainty.
- Each ‘arm’ (option) has an unknown reward distribution, and the agent must decide which arm to pull to maximise long-term reward.
- **ε-greedy strategy**
    - With probability ε, the agent explores a random arm.
    - With probability $1 - \varepsilon$, the agent exploits the arm with the highest estimated average reward.
    - Through numerous trials, the algorithm strikes a balance between discovery and performance, converging on near-optimal choices.
- **Applications:** adaptive web design (A/B testing), online advertising, recommender systems and clinical trial optimisation.

Markov decision processes (MDPs)

- MDPs extend the bandit concept to sequential decisions when each action affects future states.

- An MDP is defined by the tuple (S, A, P, R, γ), where:
  - S: set of states
  - A: available actions
  - P: transition probabilities
  - R: rewards
  - γ: discount factor
- **The Bellman equation** expresses the recursive relationship between immediate and future rewards.
- **Value iteration** repeatedly applies the Bellman update until utilities converge, producing optimal policies. In a grid world example, the agent learns to move towards the goal state while avoiding traps and penalties.

Dynamic programming

- Dynamic programming provides exact solutions when the environment model (P and R) is known. It breaks complex problems into subproblems using two main algorithms:
  - **Policy iteration:** it alternates between evaluating and refining a policy until it reaches stability.
  - **Value iteration:** it updates utilities directly using the Bellman optimality equation.
- Dynamic programming forms the foundation for planning and decision-making in RL.

Q-learning

- Q-learning is a **model-free** algorithm that learns the value of state–action pairs directly from experience.
- With sufficient exploration, Q-learning converges to the optimal action policy even when the transition model is unknown.
- **Examples** include grid world navigation, autonomous game playing (such as tic-tac-toe and Go) and robotic control.

Model-based vs model-free approaches

- Model-based methods use or learn an explicit model of environment dynamics to plan actions. They are sample-efficient and adapt quickly, but are sensitive to model inaccuracies.
- Model-free methods learn directly from rewards without modelling transitions. They are more robust but require extensive experience.

Real-world applications

RL drives innovation across many industries, including:

- **Data centres:** hierarchical RL for energy-efficient cooling and workload distribution.
- **Social media:** contextual bandits for personalised recommendations and content ranking.
- **Autonomous vehicles:** hybrid RL systems for route planning and safety optimisation.
- **Robotics:** adaptive control for walking, grasping and manipulation.
- Healthcare and finance: sequential decision-making for treatment planning or portfolio management.
- AlphaGo Zero demonstrated the power of self-play and deep RL by mastering the game of Go without human input.

Key parameters and notation

Symbol	Meaning	Typical range
α	Learning rate	$0 < \alpha < 1$
γ	Discount factor	$0.8 - 0.99$
ε	Exploration rate	$0.1 - 0.3$ (often decayed)
$Q(s, a)$	Action-value function	-
$U(s)$	State utility	-

When to use RL

Use RL when the problem involves **sequential decisions**, uncertainty and **delayed rewards**. RL is the most effective when explicit supervision is unavailable and the system must learn optimal behaviour through feedback and interaction.