Reminder: Branches of Machine Learning

- Unsupervised Learning
  - Dimensionality Reduction
  - Clustering
  - Unsupervised Learning
  - Customer Segmentation
  - Targeted Marketing
  - Recommender Systems
  - Big Data Visualization
  - Meaningful Compression

- Supervised Learning
  - Classification
  - Regression
  - Supervised Learning
  - Customer Retention
  - Diagnostics
  - Advertising Popularity Prediction
  - Weather Forecasting
  - Market Forecasting
  - Estimating Life Expectancy

- Reinforcement Learning
  - Real-time decisions
  - Game AI
  - Skill Acquisition
  - Learning Tasks
  - Robot Navigation

- Structure Discovery
  - Feature Elicitation
  - Identity Fraud Detection
  - Image Classification
  - Population Growth Prediction
Many Types and Areas of Reinforcement Learning

- Machine Learning
- Optimal Control
- Reinforcement Learning
- Operations Research
- Reward System
- Classical/Operant Conditioning
- Bounded Rationality
What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent’s actions affect the subsequent data it receives
Examples of Reinforcement Learning

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different Atari games better than humans
- Beat the best human player in Go
Examples – Helicopter Manoeuvres

https://www.youtube.com/watch?v=OJL04JJjocc
Examples – Bipedal Robots

https://www.youtube.com/watch?v=No-JwwPbSLA
A reward $R_t$ is a scalar feedback signal
- Indicates how well agent is doing at step $t$
- The agent’s job is to maximize cumulative reward

Reinforcement learning is based on the **reward hypothesis**

**Definition (Reward Hypothesis)**

All goals can be described by the maximization of expected cumulative reward

Do you agree with this statement?
Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - -ve reward for crashing
- Defeat the world champion at Backgammon
  - +/-ve reward for winning/losing a game
- Manage an investment portfolio
  - +ve reward for each $ in bank
- Control a power station
  - +ve reward for producing power
  - -ve reward for exceeding safety thresholds
- Make a humanoid robot walk
  - +ve reward for forward motion
  - -ve reward for falling over
- Play many different Atari games better than humans
  - +/-ve reward for increasing/decreasing score
Sequential Decision Making

- Goal: select actions to maximize total future reward
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
  - A financial investment (may take months to mature)
  - Refueling a helicopter (might prevent a crash in several hours)
  - Blocking opponent moves (might help winning chances many moves from now)
Agent and Environment

observation

\( O_t \)

reward

\( R_t \)

action

\( A_t \)
Agent and Environment

At each step $t$ the agent:
- Executes action $A_t$
- Receives observation $O_t$
- Receives scalar reward $R_t$

The environment:
- Receives action $A_t$
- Emits observation $O_{t+1}$
- Emits scalar reward $R_{t+1}$

$t$ increments at env. step
The **history** is the sequence of observations, actions, rewards

\[ H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \]

- i.e. all observable variables up to time \( t \)
- i.e. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- **State** is the information used to determine what happens next
- Formally, state is a function of the history:

\[ S_t = f(H_t) \]
The environment state $S_t^e$ is the environment’s private representation

i.e. whatever data the environment uses to pick the next observation/reward

The environment state is not usually visible to the agent

Even if $S_t^e$ is visible, it may contain irrelevant information
The agent state $S_t^a$ is the agent’s internal representation
i.e. whatever information the agent uses to pick the next action
i.e. it is the information used by reinforcement learning algorithms
It can be any function of the history:

$$S_t^a = f(H_t)$$
Information State

An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state $S_t$ is Markov if and only if

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, ..., S_t)$$

- The future is independent of the past given the present
  $$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- Once the state is known, the history may be thrown away
- i.e. the state is a sufficient statistic of the future
- The environment state $S^e_t$ is Markov
- The history $H_t$ is Markov
What if agent state = last 3 items in sequence?
What if agent state = counts for lights, bells and levers?
What if agent state = complete sequence?
Fully Observable Environments

Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

- Agent state = environment state = information state
- Formally, this is a Markov decision process (MDP)
Partial observability: agent indirectly observes environment:
- A robot with camera vision isn’t told its absolute location
- A trading agent only observes current prices
- A poker playing agent only observes public cards

Now agent state $\neq$ environment state

Formally this is a partially observable Markov decision process (POMDP)

Agent must construct its own state representation $S^a_t$, e.g.
- Complete history: $S^a_t = H_t$
- Beliefs of environment state: $S^a_t = (P(S^e_t = s^1), \ldots, P(S^e_t = s^n))$
- Recurrent neural network: $S^a_t = \sigma(S^a_{t-1} W_s + O_t W_o)$
An RL agent may include one or more of these components:

- **Policy**: agent’s behaviour function
- **Value function**: how good is each state and/or action
- **Model**: agent’s representation of the environment
A **policy** defines the agent’s behaviour

- It is a map from state to action, e.g.
- Deterministic policy: \( a = \pi(s) \)
- Stochastic policy: \( \pi(a|s) = P(A_t = a|S_t = s) \)
Value Function

- **Value** function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- Used to select which action to take
- e.g. models the **expected discounted future** reward

\[
\nu_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s]
\]


- A **model** predicts what the environment will do next
- \( \mathcal{P} \) predicts the next state
- \( \mathcal{R} \) predicts the next (immediate) reward, e.g.

\[
\mathcal{P}_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a) \\
\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]
\]

- Can be used for planning without actually performing actions
Maze Example

Rewards: $-1$ per time-step
Actions: N, E, S, W
States: Agent’s location
End at Goal state
Maze Example: Policy and Value Function

Arrows represent policy $\pi(s)$

Numbers represent value $v_\pi(s)$
Maze Example: Model

Agent may have an internal model of the environment

Dynamics: how actions change the state

Rewards: how much reward from each state

The model may be imperfect (most likely is)

Grid layout represents transition model \( P_{ss'}^a \)

Numbers represent immediate reward \( R_s^a \) from each state \( s \) (same for all \( a \))
Categorization of RL agents

- **Value Based**
  - No Policy (Implicit)
  - Value Function

- **Policy Based**
  - Policy
  - No Value Function

- **Actor Critic**
  - Policy
  - Value Function

- **Model Free**
  - Policy and/or Value Function
  - No Model

- **Model Based**
  - Policy and/or Value Function
  - Model
Learning and Planning

Two fundamental problems in sequential decision making

- **Reinforcement Learning:**
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy
  - a.k.a. learning by doing, trail and error learning

- **Planning:**
  - A model of the environment is known
  - The agent performs computations with its model (without any external interaction)
  - The agent improves its policy
  - a.k.a. deliberation, reasoning, introspection, pondering, thought, search
Atari Example: Reinforcement Learning

- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores
Atari Example: Planning

- Rules of the game are known
- Can query emulator
  perfect model inside agent’s brain
- If I take action $a$ from state $s$:
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  e.g. tree search
Reinforcement learning is like trial-and-error learning
The agent should discover a good policy
From its experiences of the environment
Without losing too much reward along the way

Exploration finds more information about the environment
Exploitation exploits known information to maximize reward
It is usually important to explore as well as exploit
Examples

- Restaurant Selection
  - **Exploitation**: Go to your favorite restaurant
  - **Exploration**: Try a new restaurant

- Online Banner Advertisements
  - **Exploitation**: Show the most successful advert
  - **Exploration**: Show a different advert

- Game Playing
  - **Exploitation**: Play the move you believe is best
  - **Exploration**: Play an experimental move

- Robot Control
  - **Exploitation**: Do the movement you know works best
  - **Exploration**: Try a different movement
Subproblems within Reinforcement Learning:

- **Prediction**: evaluate the future
  How do I do given a policy?
- **Control**: optimize the future
  Find the best policy
Gridworld Example: Prediction

-1 in each step

Teleport from A to A' and get 10, B to B' and get 5

What is the value function for the uniform random policy?
Gridworld Example: Control

What is the optimal value function over all possible policies?
What is the optimal policy?
Markov Decision Processes
Markov Process

A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

**Reminder: Markov property**

A state $S_t$ is Markov if and only if

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)$$

**Definition (Markov Process/ Markov Chain)**

A *Markov Process* (or *Markov Chain*) is a tuple $(S, P)$

- $S$ is a (finite) set of states
- $P$ is a state transition 0 probability matrix,

$$P_{ss'} = P(S_{t+1} = s' \mid S_t = s)$$
Example: Student Markov Chain

[Diagram of a Markov chain with states labeled as Facebook, Sleep, Class 1, Class 2, Class 3, and Pub. Arrows indicate transitions between states with transition probabilities labeled on the arrows.]
Example: Student Markov Chain Episodes

Sample episodes for starting from $S_1 = \text{C1}$  

$S_1, S_2, \ldots, S_T$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep
Example: Student Markov Chain Transition Matrix

\[ \mathcal{P} = \begin{bmatrix}
C1 & C2 & C3 & Pass & Pub & FB & Sleep \\
C1 & 0.5 & & & & & \\
C2 & & 0.8 & & & & \\
C3 & & & 0.6 & 0.4 & & \\
Pass & & & & & 0.2 & \\
Pub & & & & 0.4 & 0.4 & \\
FB & & & & 0.1 & & 0.9 \\
Sleep & & & & & & 1
\end{bmatrix} \]
A Markov reward process is a Markov chain with values.

**Definition (MRP)**

A *Markov Reward Process* is a tuple \((S, \mathcal{P}, \mathcal{R}, \gamma)\)

- \(S\) is a finite set of states
- \(\mathcal{P}\) is a state transition probability matrix,
  \[
  P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)
  \]
- \(\mathcal{R}\) is a reward function, \(\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]\)
- \(\gamma\) is a discount factor, \(\gamma \in [0, 1]\)
Example: Student MRP

![Diagram of a Markov Reward Process](image-url)
The return $G_t$ is the total discounted reward from time-step $t$.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The discount $\gamma \in [0, 1]$ is the present value of future rewards.
- The value of receiving reward $R$ after $k + 1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - $\gamma$ close to 0 leads to “myopic” evaluation.
  - $\gamma$ close to 1 leads to “far-sighted” evaluation.
Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward
- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.
Next time

- Continue with MDP’s and Bellmann Equation
- Dynamic Programming and Q-Learning