Overview
What is RL and how does it work?

- So general and interesting at an intersection of several fields (computer science, engineering, neuroscience, mathematics, economics, and psychology) to understand the optimal way to make decisions, connected to how and why make decisions if they are trying to optimize utility
- Characteristics of RL:
    - No supervisor (no one telling the right action to take, instead a trial and error paradigm - only a reward signal)
    - Feedback is delayed (retrospectively see if a decision good or bad in later steps), not instantaneous
    - Time really matters (sequential decision-making processes in a dynamic system)
        - Agent's actions influence the subsequent data it receives
- RL Problem:
    - Reward (R_t): scalar feedback signal - how well agent is doing at time step *t*
    - Agent's goal is to maximize reward in total
    - RL is based on premise of *reward hypothesis*: all goals can be described by the maximisation of expected cumulative reward
    - Unifying framework of *sequential decision making*: goal is to select actions to maximize total future award
        - Actions could have long-term consequence or reward might be delayed, meaning it could be better to sacrifice immediate reward to gain more long-term reward
        - Balance immediate vs. long-term reward
- Agent and Environment:
    - Agent: executes actions (A_t) based on observation (O_t - information of environment) and receives reward signal (R_t - how well it's doing in that step)
    - Environment: each step/loop, environment emits observation and reward and receives action
- History and State:
    - Time series of observation, reward, and action → experience data stream → *history* ($H_t = A_1, O_1, R_1, \dots A_t, O_t, R_t$): sequence of observation, action, rewards (all observable variables up to time *t*)
        - Agent selects actions from history
        - Environment looks at what it's received to select observations/rewards
    - *State*: summary of information used to determine what happens next
        - Function of history $S_t = f(H_t)$
        - Different definitions:

                - *Environment state* (S^e_t): information used within the environment to determine the next observation and reward - environment's private representation → not usually visible to the agent (even when visible, may contain irrelevant info)
                - *Agent state* (S^a_t): agent's internal representation of information used to determine next action; information used by RL algorithms - our decision to how to process and what to remember/what to throw away
                    - Could be any function of history ($S^a_t = f(H_t)$) compiled to useful information to characterize its future behavior
                - *Information state* (aka *Markov state*): mathematical definition - contains all useful information from history
                    - State S_t is Markov iff Markov property where probability of next state (S_{t+1}) conditioned on current state (S_t) = probability of next state given all previous states ($S_1, \ldots, S_t$) → can just retain current state to get same characterization of future
                        - Future is independent of the past given the present
                        - The state is a sufficient statistic of the future
                    - By definition, environment state S^e_t and entire history H_t are Markov (⇒ there's always a Markov state → how do we find the useful representation)
                - Multiple options for characterization of agent state, which defines what happens next and our job is to define useful representation that effectively predicting what happens next
- Environment visibility:
    - Fully observable environment: agent directly observes environment state
        - $O_t = S^a_t = S^e_t$ → Markov state = agent state = env. state
        - Formally, this is a MDP
    - Partially observable environment: agent indirectly observes env.
        - Need to build an agent state distinct from env. state (unknown)
        - Formally, this is POMDP (partially observable MDP)
        - Agent must construct its own state representation S^a_t via
            - Complete history ($S^a_t = H_t$)
            - Beliefs of env. state: Bayesian probability distribution over where you think you are in the env. (vector of probabilities of different env. state options defines the state used to decide action)
            - RNN (not probability): take agent state from prev. state and take linear combination with latest observation → new state
- Inside an RL Agent

    - Potential components of RL agent:
        - Policy: how agent picks action → agent's behavior function
            - Map from state to action
            - Deterministic policy: a = pi(s)
            - Stochastic policy: pi(a|s) = probability of taking particular action given some states
        - Value function: how good is each state and/or action (how much expected reward)
            - Prediction of expected future reward from some state onward if we continue particular behavior (policy)
            - Used to evaluate goodness/badness of states → used to compare different behavior options
            - Future reward estimates are discounted
                - $v_{pi}(s) = E_{pi}[R_t + gamma*R_{t+1} + gamma^2*R_{t+2} \dots | S_t = s]$
        - Model: agent's representation of the environment
            - Predicts what environment will do next
            - 2 parts to model:
                - Transition model: *P* predicts next state (dynamics of environment) - probability of next state *s'* given previous state and action
                - Reward model: *R* predicts next (immediate) reward - probability of reward given previous state and action
- Taxonomy of RL Agents (according to three components)
    - Value based algorithm:
        - Value function
        - Implicit policy (look at value function and choose best action)
    - Policy based algorithm:
        - Policy (stores policy and adjust policy for most possible reward)
        - Implicit value function
    - Actor critic: stores both policy and value function
    - Model free
        - Policy and/or value function (see experience to behave for most reward)
        - No explicit understanding of environment (model)
    - Model based
        - Policy and/or value function
        - Model (firstly build up model of how env. works to determine how to behave for most reward)
- Problems within RL
    - 2 fundamental problems in sequential decision making
        - Reinforcement learning problem:

        - Environment is initially unknown
        - Agent interacts with environment (trial and error learning)
        - Agent improves policy to maximize future reward
    - Planning:
        - Model of env. is known
        - Agent performs internal computations with model (without external interaction)
        - Agent improves policy by planning ahead (e.g. tree search)
- Balancing exploration and exploitation (unique to RL)
    - Agent should discover optimal policy from its experience of the env. without losing too much reward along the way
    - Exploration: give up known reward to find out more about the env. (finds more info about the env.)
    - Exploitation: exploit known information to maximize reward
- Prediction and control
    - Prediction: evaluate future given current policy - optimal value function across all possible policies
    - Control: optimize future by finding the best policy - optimal policy
    - In RL, need to solve prediction problem to solve control problem