

Reinforcement Learning

Lecture 1: Introduction to Reinforcement Learning

Max(Chong) Li

Outline

- Admin
- Introduction to reinforcement learning
- Elements of reinforcement learning
- Reinforcement learning problem and example
- History

Admin

- Instructor: Dr. Max(Chong) Li, Dr. Chonggang Wang
- Email: cl3607@columbia.edu, cgwang@ieee.org
- Office hour: TBD
- TA: Trevor Joseph Gordon (tjg2148@columbia.edu)
- CA: Gokul Srinivasan (gs3160@columbia.edu)
 - Jinxuan Tang (jt3302@columbia.edu)
 - Xin Huang (xh2510@columbia.edu)
 - Yukai Song (ys3493@columbia.edu)

Admin

- Textbook: “Reinforcement Learning: An Introduction”, R. Sutton and A. Barto, (2nd edition)
- References:
 - “Reinforcement Learning for Cyber Physical Systems with Cybersecurity Case Study”, 2019, C. Li and M-K Qiu
 - “Algorithms for Reinforcement Learning”, Szepesvari, (available online)

Prerequisites

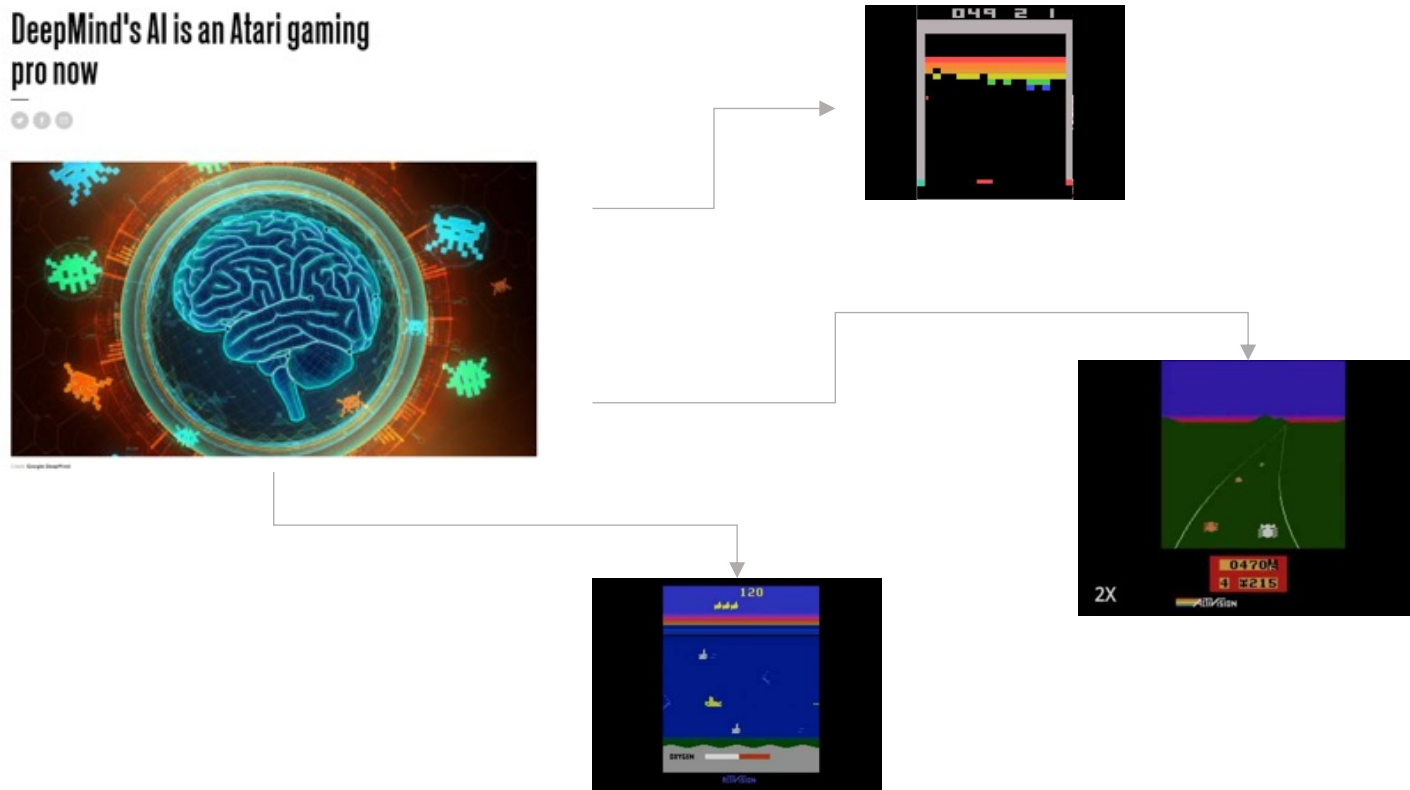
- Elementary statistics & Probability theory
 - Expectation, variance, distribution, ...
- Basic linear algebra
 - Vectors, matrix operations, gradients, eigenvalue, ...
- Basic programming skills
 - Proficiency in Python,
 - Knowledge of tensorflow and openAI gym
- Background on machine learning is a plus
- Your passion and interest

Grading

- Bi-weekly Assignments: 50%
 - Including 1 coding assignment
- Midterm exam: 20%
- Final exam: 30%

Significant RL Success

- Achieved human-level performance on Atari games from pixel-level visual input, in conjunction with deep learning (Google DeepMind 2015)



Significant RL Success

- AlphaGo was developed by Google DeepMind in London in October 2015
- It became the first Computer Go program to beat a human professional Go player on a full-sized 19×19 board. In March 2016, it beat Lee Sedol in a five game match
- AlphaGo Zero (2017) a new version created without using human input data and stronger than any previous version, its successor AlphaZero is currently perceived as the world's top player in Go.



What is RL?

- Agent-oriented learning—learning by interacting with an environment to achieve a goal
 - more realistic and ambitious than other kinds of machine learning
- Learning by **trial and error**, with only delayed evaluative feedback (reward)
 - the kind of machine learning most like natural learning
 - learning that can tell for itself when it is right or wrong

Characteristics of RL

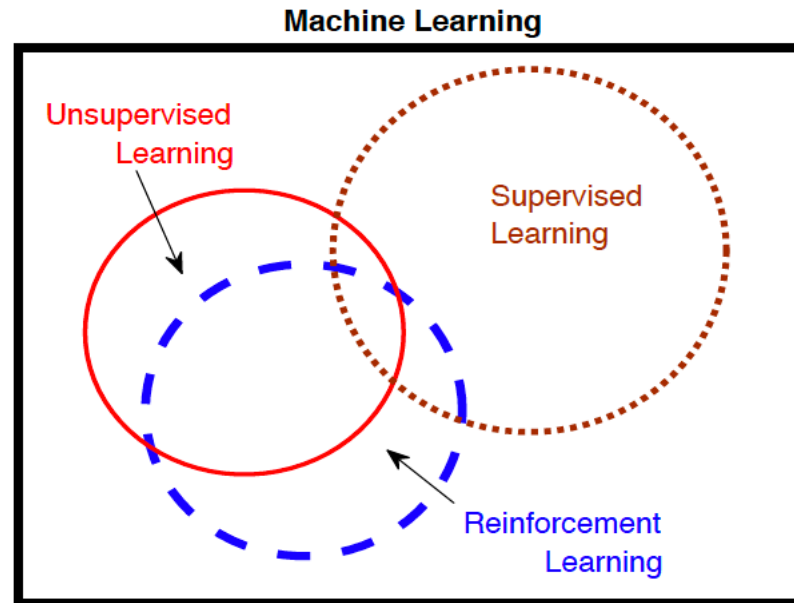
- What makes RL different from other machine learning paradigms?
 - No supervisor, only a reward signal
 - Feedback is delayed, not instantaneous
 - Time matters, i.e., sequential decision making
 - Agent's actions affect the subsequent data/feedback from the environment



Applications of RL

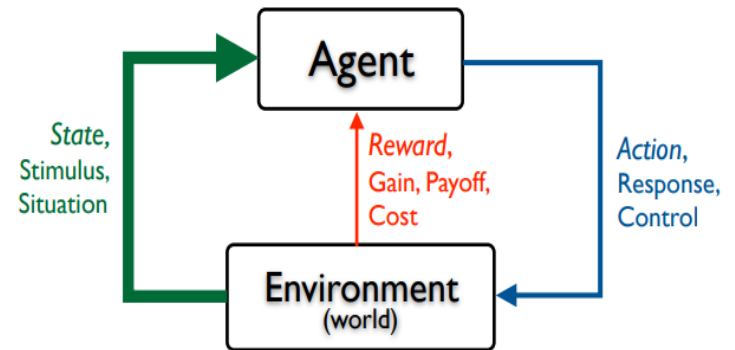
- Manufacturing:
 - Sort products in eCommerce and supermarkets
 - Assembling line: Tesla's factory
- Inventory Management
 - Reduce transit time for stocking/retrieving products
- Delivery Management
 - Delivery vehicle routing
- Power systems
 - Assess the security and enhance microgrid performance
- Finance
 - Evaluate trading strategies

Branches of Machine Learning



Basic RL Interface

- History and State
 - history is the sequence of observations, actions and rewards
 - state is the information used to determine what happens next. It is a function of the history
- At each time step, an agent
 - executes action
 - receives observation of the environment
 - receives reward from the environment
- At each time step, the environment
 - receives action from the agent
 - emits observation/changes its own states
 - emits reward to the agent



Elements of RL

- Policy

- A policy, π , is a mapping from perceived states of the environment to (the probability of) actions $a \in \mathcal{A}(s)$ to be taken when in those states.

$$s \in \mathcal{S} \longrightarrow \pi(a|s)$$

- the core of RL to determine behavior
- Policy can be stochastic or deterministic

- Reward and Return

- Reward is a mapping from each perceived state of the environment to a single number, indicating the intrinsic desirability of that state
- Reward is immediate
- Return is a cumulative sequence of received rewards after a given time step
- Finite step return:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

- Discounted return: $0 \leq \gamma \leq 1$.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Elements of RL

- Value function

- Functions of states (or of state-action pairs) that estimate *how good* it is for the agent to be in a given state
- “how good” refers to the expected return
- For Markov Decision Process (MDP), the value of a state is defined formally as

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

- For MDP, the value of an action-state pair is defined formally as

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

- Model (optional)

- Mimic the behavior of the environment
- Used for planning (decide on a course of actions by considering future situations before experienced)
- MDP model:

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

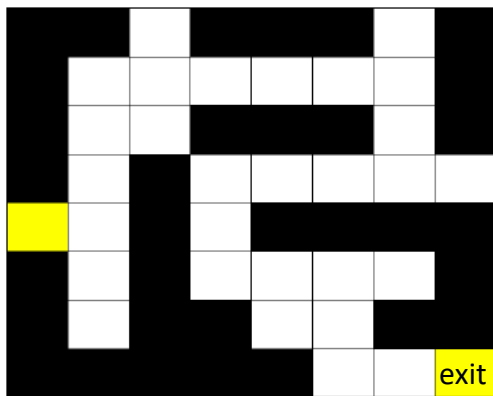
$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

What is RL problem?

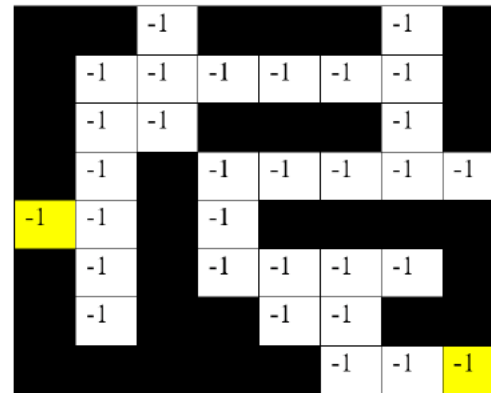
- RL problem is a considerable abstraction of the problem of goal-directed learning from interaction with the environment
- RL methods/solutions specify how the agent changes its policy as a result of its experience.
- The agent's goal is to **maximize** the total amount of reward it receives over the long run

Maze Example: Model & Reward

- The internal model of the environment may or may not be given to the agent
- Actions: N, E, S, W
- States: Agent's location
- Reward: -1 per step (why?)

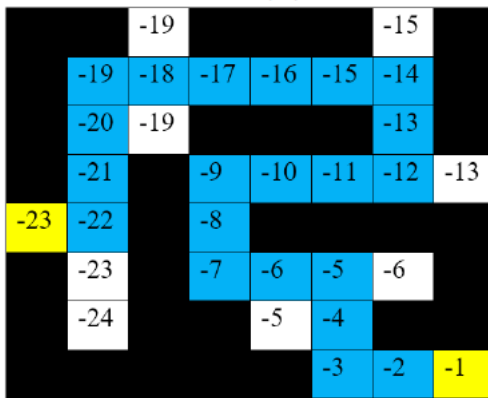


Model: Maze Map

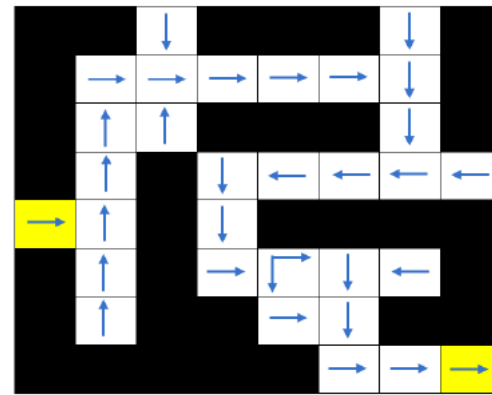


Reward

Maze Example: Value Function & Policy



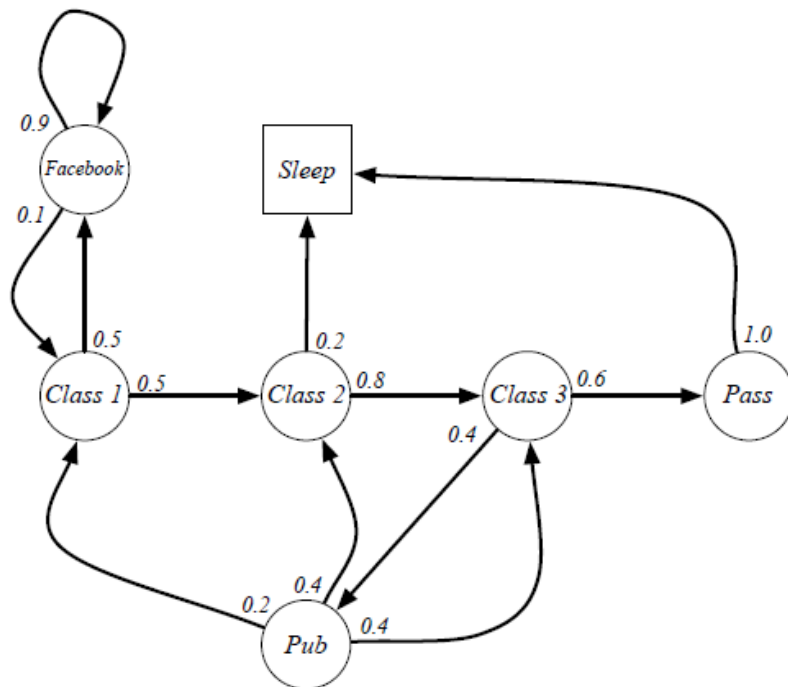
- Numbers represent value of each state



- Arrows represent policy (action) for each state

Example: Student Markov Chain*

- Stochastic policy:



Sample **episodes** for Student Markov Chain starting from $S_1 = C1$

$$S_1, S_2, \dots, S_T$$

- 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

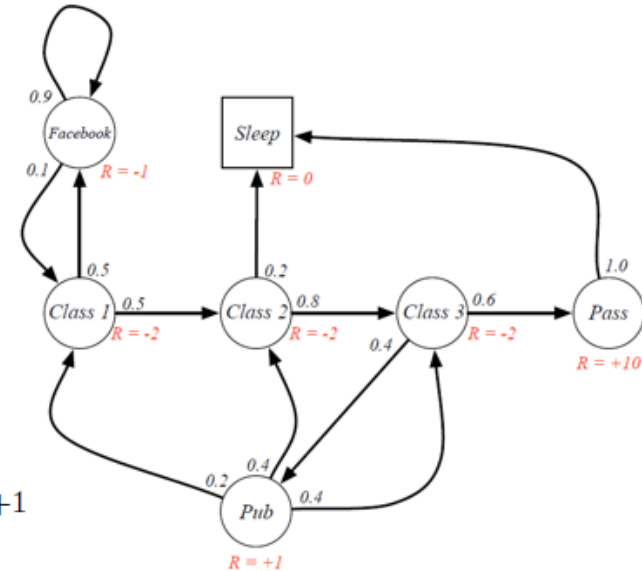
* This example is taken from David Silver's lecture notes

State-Value Function Evaluation

Sample **returns** for Student MRP:
Starting from $S_1 = C1$ with $\gamma = \frac{1}{2}$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	=	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	=	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	=	-3.41
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	=	-3.20
FB FB FB C1 C2 C3 Pub C2 Sleep			



The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

MDP

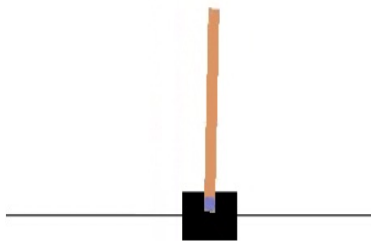
- A Markov decision process (MDP) is an environment in which **all states are Markov**
- RL essentially solves a MDP problem.

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

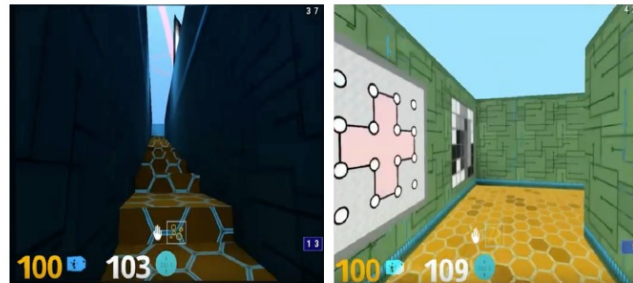
- \mathcal{S} is a finite set of states
- \mathcal{A} is a finite set of actions
- \mathcal{P} is a state transition probability matrix,
 $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- \mathcal{R} is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

RL Simulation Toolkits

- To evaluate RL algorithms in simulations, need to first create an environment and the agent-environment interface
- Environment can be very complicated
- Widely used toolkits as a collection of environments designed for testing
 - OpenAI Gym: simple games/environment from walking to playing “pong”
 - OpenAI Universe: platform across the world’s supply of games and websites.
 - DeepMind Lab: 3D navigation and puzzle-solving environment



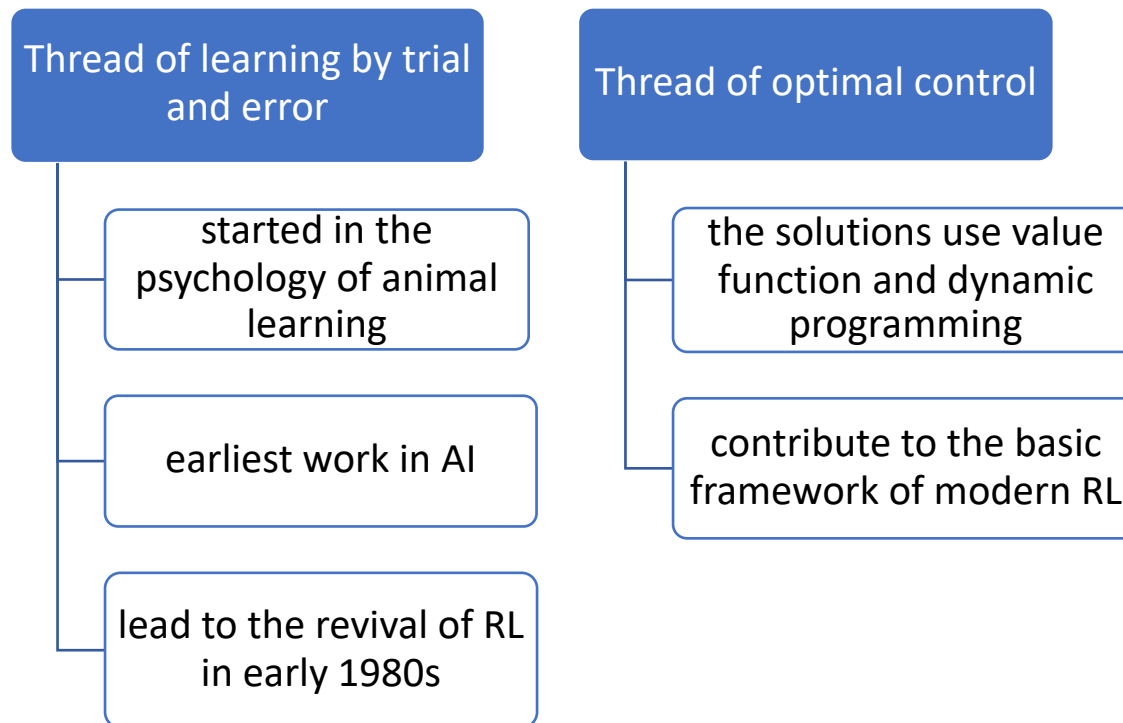
OpenAI Gym: CartPole



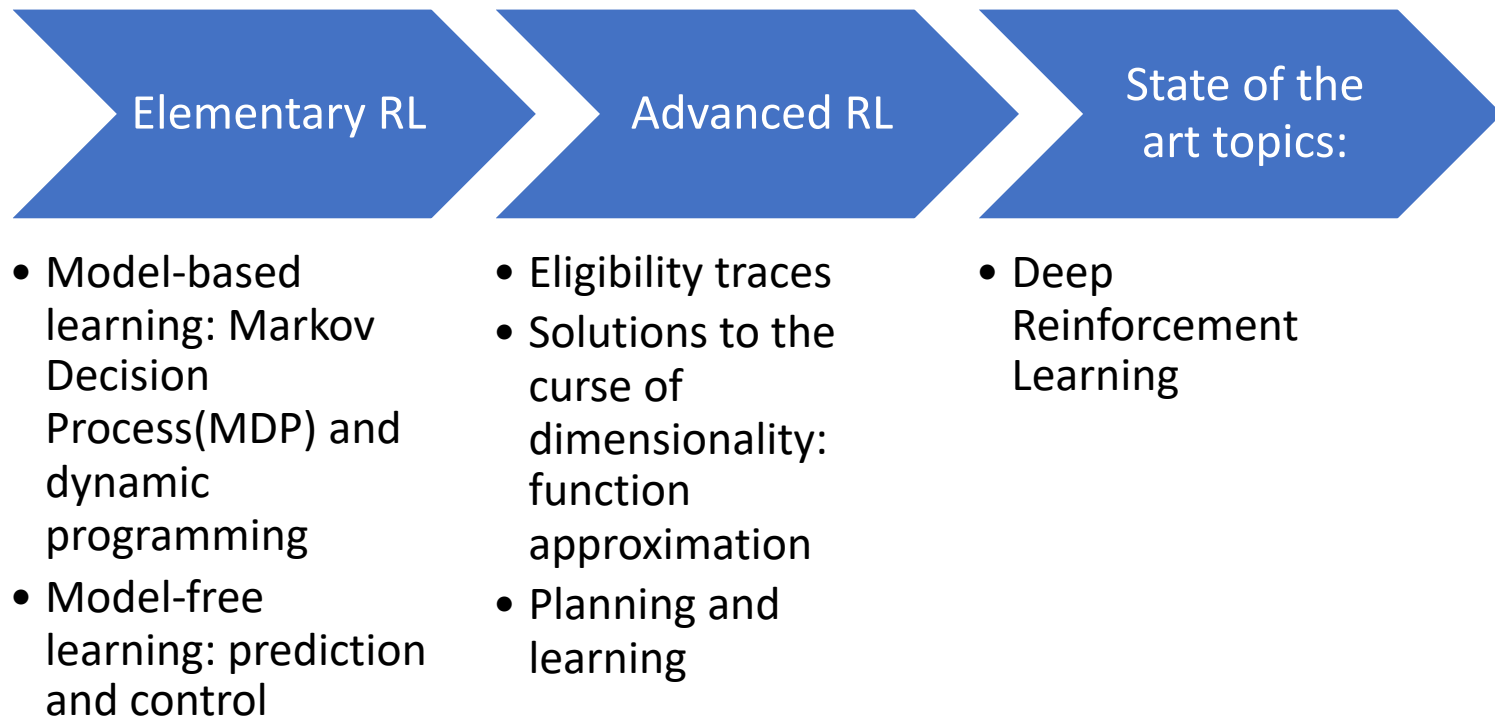
DeepMind Lab: 3D Maze

History of RL

- Two main threads to the modern RL theory. Developed independently and then merged in late 1980s



Course Outline



Google's Deep Q-learning playing Atari Breakout

