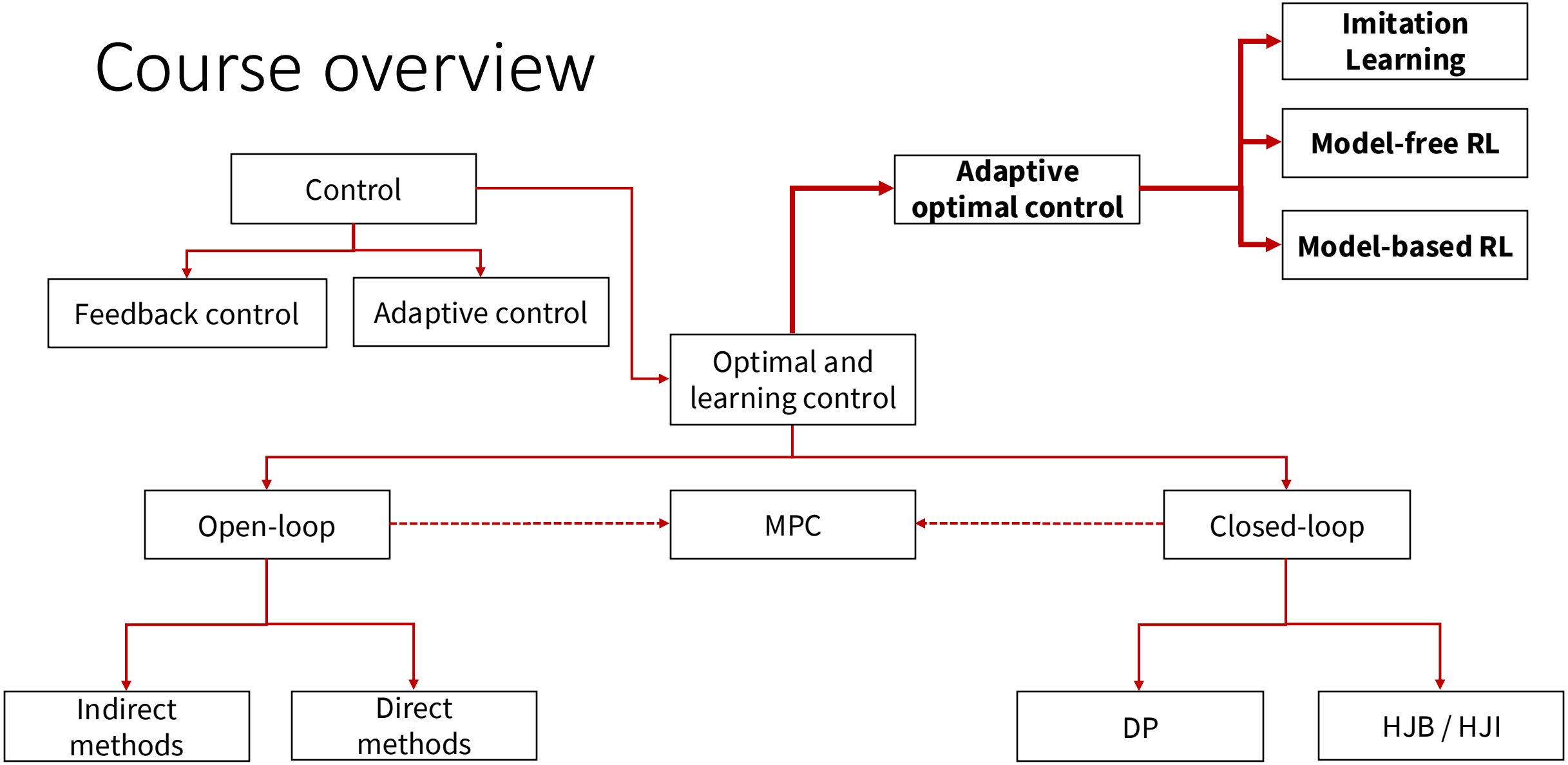
AA203
Optimal and Learning-based Control

Intro to Imitation Learning (IL) and Reinforcement Learning (RL)

Stanford University

ASL
Autonomous Systems Lab

Course overview



Why study learning for control?

12 DoF quadruped

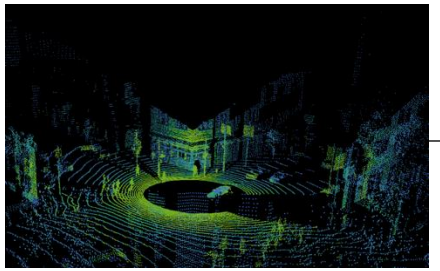
Lidar

RGB front camera

Potentially hazardous terrain

…

How can we develop a reliable control algorithm?



Upcoming weeks: Set it up as a *learning problem*

Option 1

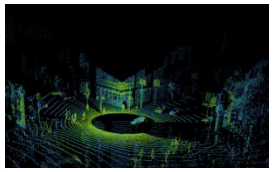
Deeply understand the problem

Design a solution

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ & \mathbf{u}(t) \in \mathcal{U} \end{aligned}$$

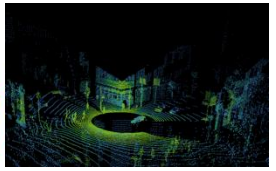
Option 2

"Learn from demonstrations"

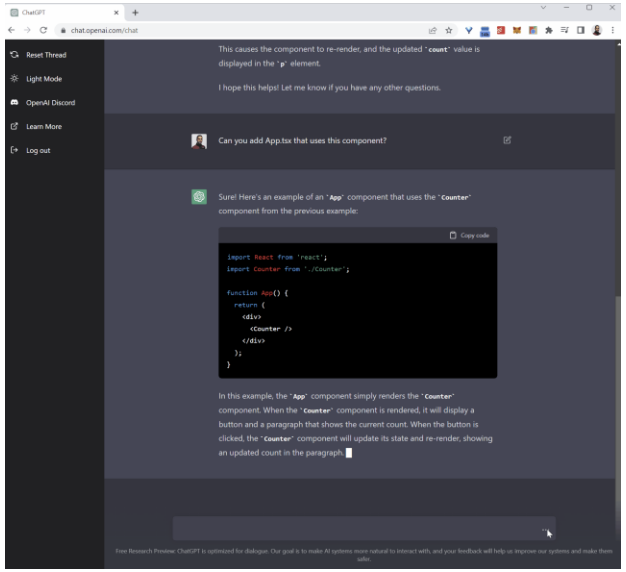
$\{$  ; $\mathbf{u}^*(t)$ $\}$

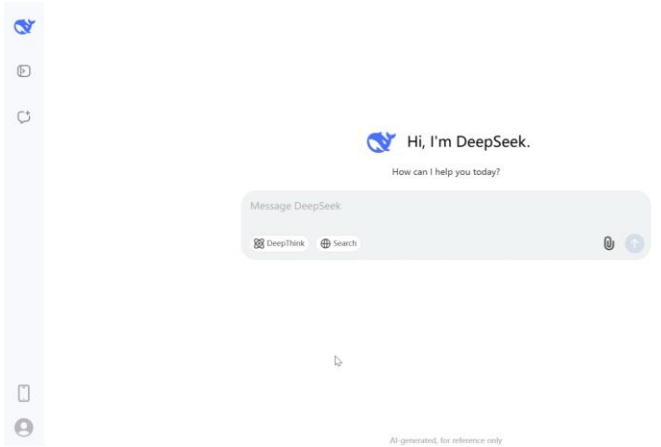
Option 3

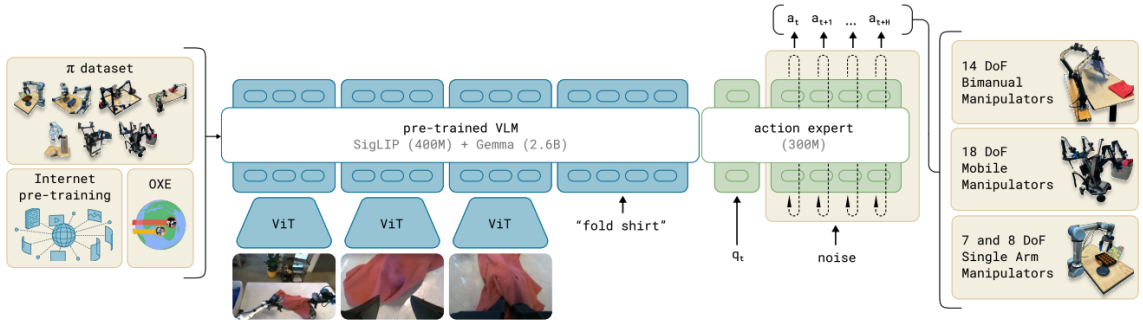
"Learn by trial and error"

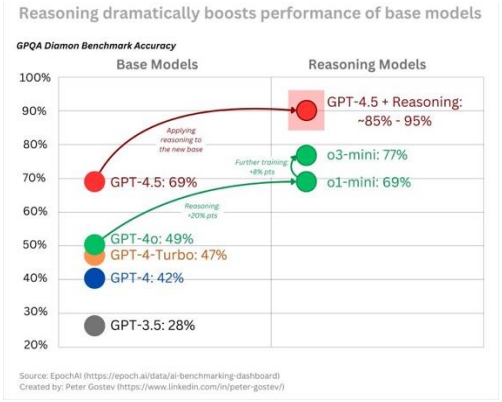
$\{$  ; Success, Failure $\}$

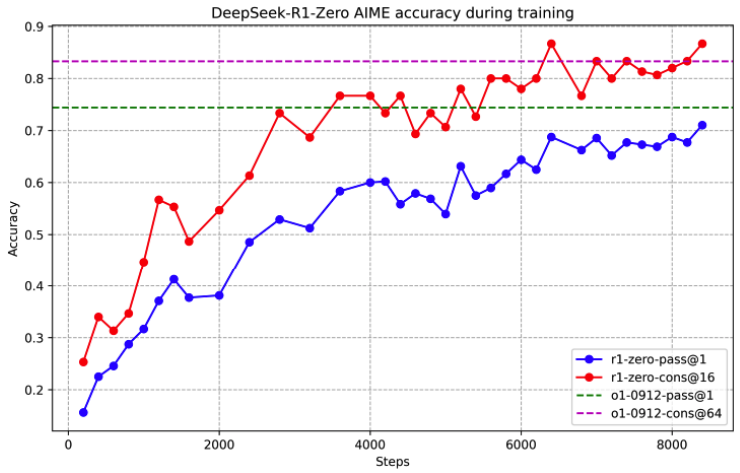
Why study learning for control?









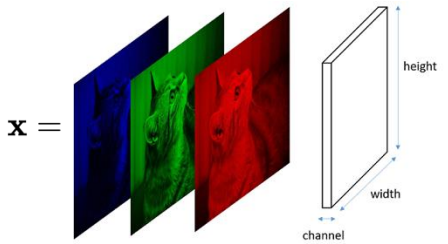


Outline

Primer on Supervised Learning

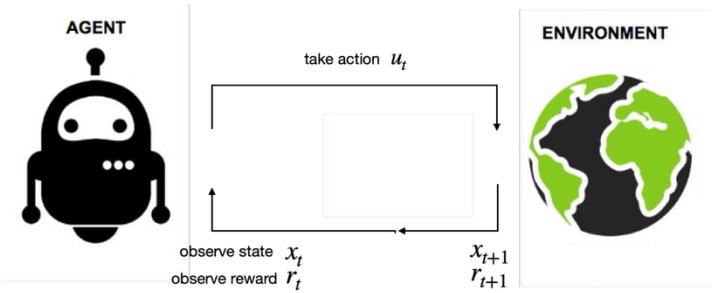
Imitation Learning

Reinforcement Learning



$f : \mathcal{X} \to \mathcal{Y}$ $\mathcal{Y} = \{0, 1, \ldots, C\}$

`e.g., [dog, cat, …, mouse]`



$$\tau = (x_0, u_0, \ldots, x_N, u_N)$$

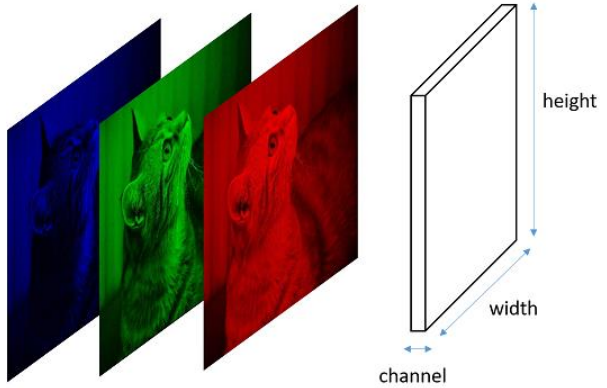
Key learning goals:

- An overview of the fundamentals of IL and RL
- Identify the distinctions and similarities between these paradigms

A primer on supervised learning (SL)

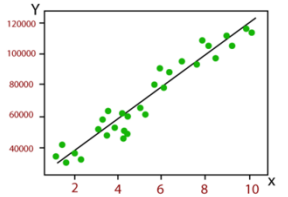
- In SL, the task is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$

  from inputs $\mathbf{x} \in \mathcal{X}$ to outputs $\mathbf{y} \in \mathcal{Y}$

$\mathbf{x} =$ 

$f : \mathcal{X} \to \mathcal{Y}$

$f : \mathcal{X} \to \mathcal{Y}$

$\mathcal{Y} = \mathbb{R}^n$ 

$\mathcal{Y} = \{0, 1, \ldots, C\}$

`e.g., [dog, cat, …, mouse]`

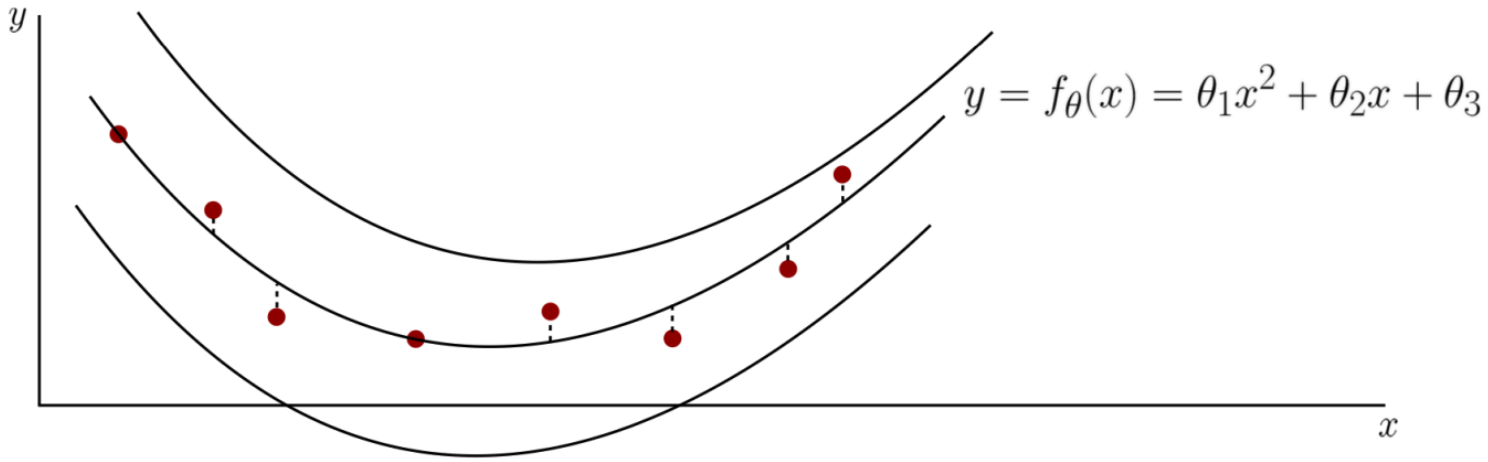
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_D \end{bmatrix}$$

- Learning involves data in the form of input-output pairs $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ e.g., Maximum Likelihood Estimation

$$\text{NLL}(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \log p(\mathbf{y} | f_\theta(\mathbf{x})) \quad \hat{\boldsymbol{\theta}}_{\text{mle}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \text{NLL}(\boldsymbol{\theta})$$

Learning from a probabilistic standpoint

- Let's consider regression as an example

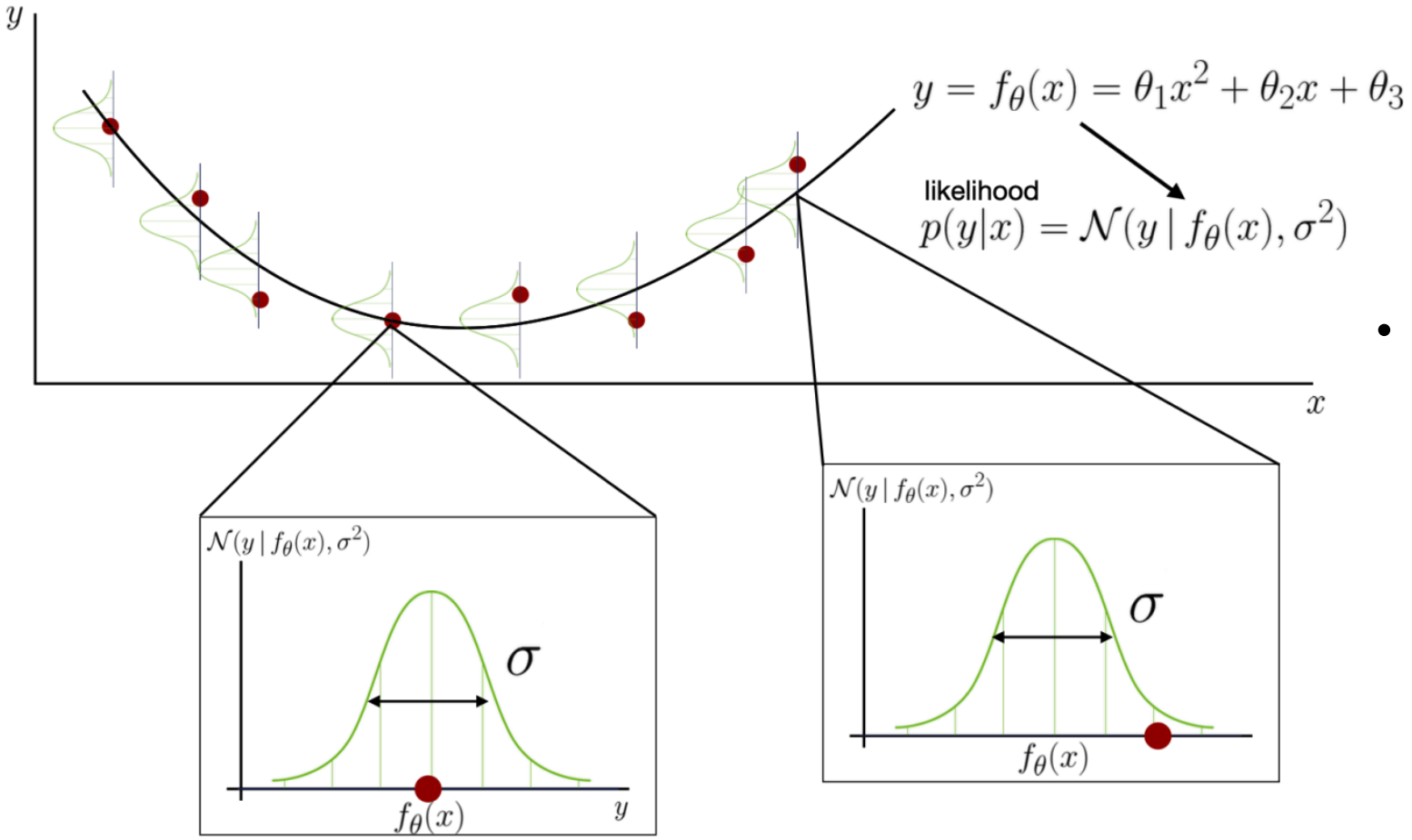


- Learning through minimization of squared error

$$\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2$$

Learning from a probabilistic standpoint

- Let's consider regression as an example



- Learning through likelihood maximization:

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} \mathcal{N}(y_i \mid f_\theta(x_i), \sigma^2)$$

$$\mathbf{NLL}(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \log p(\mathbf{y}|f_\theta(\mathbf{x}))$$

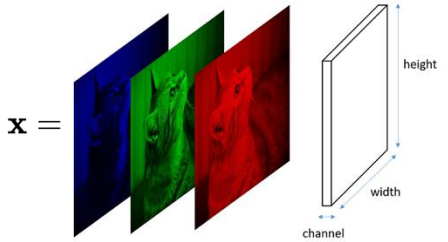
$$\hat{\boldsymbol{\theta}}_{\mathbf{mle}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbf{NLL}(\boldsymbol{\theta})$$

Outline

Primer on Supervised Learning

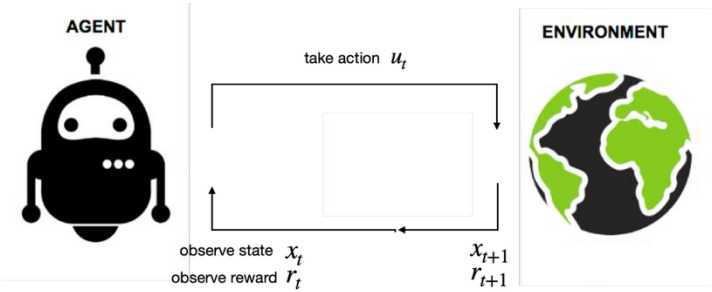
Imitation Learning

Reinforcement Learning



$f : \mathcal{X} \to \mathcal{Y}$ $\mathcal{Y} = \{0, 1, \ldots, C\}$

`e.g., [dog, cat, …, mouse]`



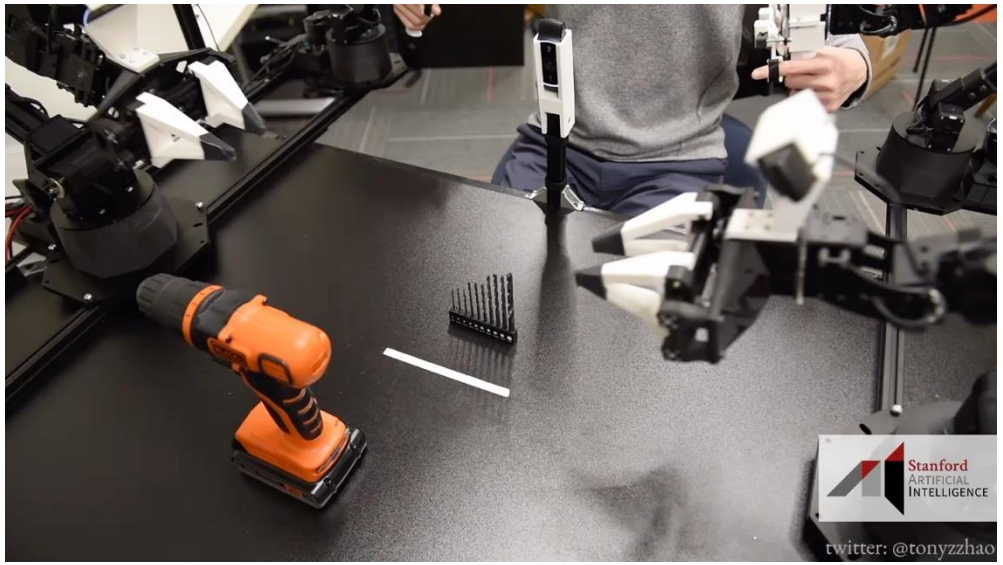
$$\tau = (x_0, u_0, \ldots, x_N, u_N)$$

Key learning goals:

- An overview of the fundamentals of IL and RL
- Identify the distinctions and similarities between these paradigms

The basics of Imitation Learning

- Imitation learning refers to a class of methods that enable skills to be transferred from an *expert* to a *learner*
- In the context of robotics, the expert is typically a human operator or a pre-existing control policy and the learner is the robot that aims to mimic the expert's behavior
- *Example:*



The basics of Imitation Learning

- Assume access to a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{u}_n)\}_{n=1}^N$ of state-control pairs collected by an expert
- At a high-level, approaches to IL belong to two main categories:
  - *Behavior Cloning* — *"learn the policy used by expert"*
  - *Inverse Reinforcement Learning (a.k.a. Inverse Optimal Control)* — *"learn the objective optimized by the expert"*

Behavior Cloning

$$\pi_\theta(\mathbf{u}|\mathbf{x})$$

Inverse Reinforcement Learning

$$R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$$

Behavior Cloning (BC)

Main idea: Train policy via supervised learning

Skeleton of an IL algorithm:

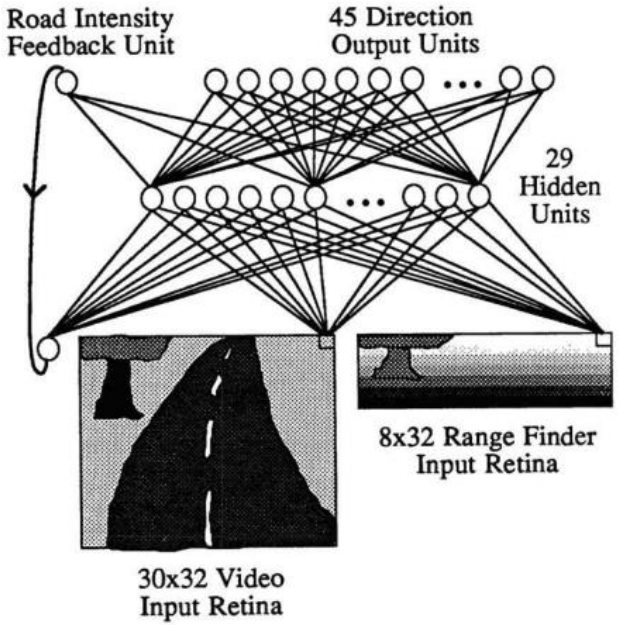
1) Collect a dataset of "expert" demonstrations  $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{u}_n)\}_{n=1}^{N}$

2) Train policy $\pi_\theta(\mathbf{u}|\mathbf{x})$ to mimic expert: $\mathrm{NLL}(\theta) = -\frac{1}{N}\sum_{n=1}^{N} \log \pi_\theta(\mathbf{u}|\mathbf{x})$

e.g., MLE, cross-entropy, L2 / MSE, etc. $\hat{\boldsymbol{\theta}}_{\mathrm{mle}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathrm{NLL}(\boldsymbol{\theta})$

A first (deep) imitation learning system

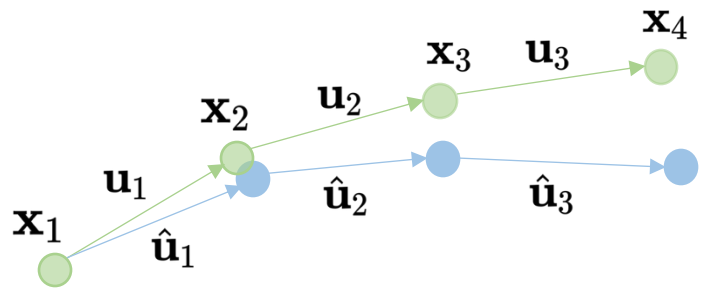
ALVINN: **A**n **A**utonomous **L**and **V**ehicle **I**n a **N**eural **N**etwork (1989)



Is this guaranteed to work?

Spoiler: NO - Imitation learning is different from “plain” supervised learning

The learner may encounter situations not represented in the expert's demonstrations



Inverse Reinforcement Learning (IRL)

IRL takes an orthogonal approach to IL, by attempting to recover a reward function from a policy, or from demonstrations of a policy

Example: (BC vs IRL)
A warehouse robot must navigate from a starting point to a goal while avoiding obstacles.
Behavior Cloning:

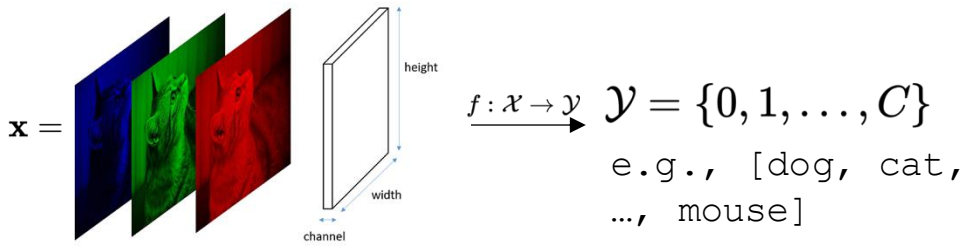
- Collect demonstration data, train a policy that imitates the demonstrations
- Effective in familiar scenarios, however, it may struggle if the layout changes substantially

Inverse RL:

- Collect the same demonstration data, infer the reward function (e.g., positive for moving closer to the goal, negative for moving close to obstacles)
- The reward function encapsulates the underlying principles of the task

Supervised Learning vs Imitation Learning

- Supervised learning: learn a mapping from input data to output labels based on a dataset of input-output pairs



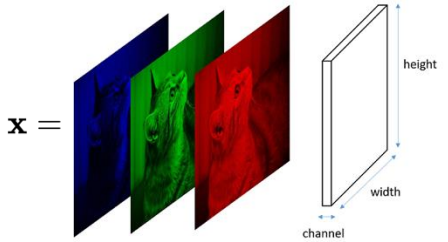
- IL builds heavily on SL, however with some key differences:
  - The solution may have inherent structural properties, such as physical constraints or temporal dependencies, that are not present in standard supervised learning tasks
  - In IL, the source domain may differ from the one at deployment (e.g., humanoid control by learning from videos of humans)
  - Covariate shift, where the distribution of the expert's data may differ from the distribution of the learner's data
  - Obtaining expert demonstrations may be expensive or time consuming

Outline

Primer on Supervised Learning

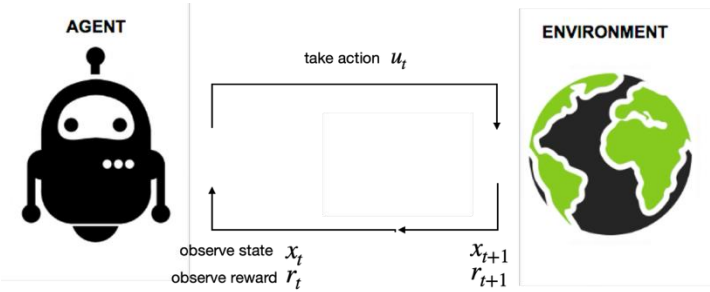
Imitation Learning

Reinforcement Learning



$f : \mathcal{X} \to \mathcal{Y}$ $\mathcal{Y} = \{0, 1, \ldots, C\}$

`e.g., [dog, cat, …, mouse]`



$$\tau = (x_0, u_0, \ldots, x_N, u_N)$$

Key learning goals:

- An overview of the fundamentals of IL and RL
- Identify the distinctions and similarities between these paradigms

What is reinforcement learning?

Fundamentally:

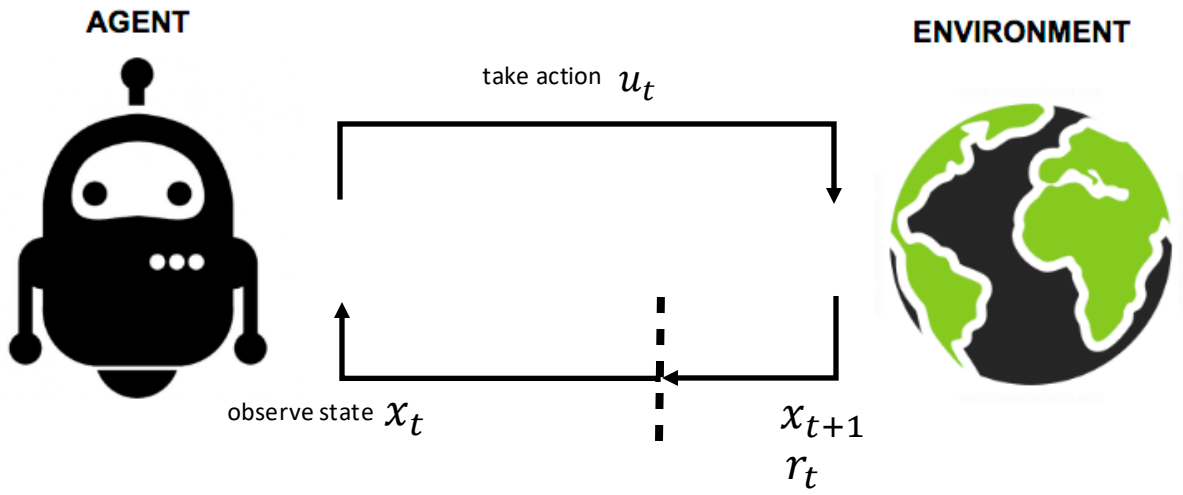
- A mathematical formalism for **learning-based** decision making
- An approach for learning decision making and control **from experience**

Elements of RL:

- **Policy** $\pi(u_t \mid x_t)$:a mapping from states to actions that defines the agent's behavior
- **Environment:** the system the agent interacts with. Mathematically, we represent it as $\mathrm{p}(x_{t+1} \mid x_t, u_t)$
- **Reward** $\mathrm{R}(x_t, u_t)$:scalar measure of (immediate) success. Defines the goal of the agent
- **Value function:** measures performance in the long run. Defines how much the agent can expect to achieve in the future
- **Model (optional):** represents the agent's understanding of the environment. Its goal is to mimic the behavior of the environment. Mathematically, we represent it as $\mathrm{p}(\hat{x}_{t+1} \mid x_t, u_t)$

What is reinforcement learning?

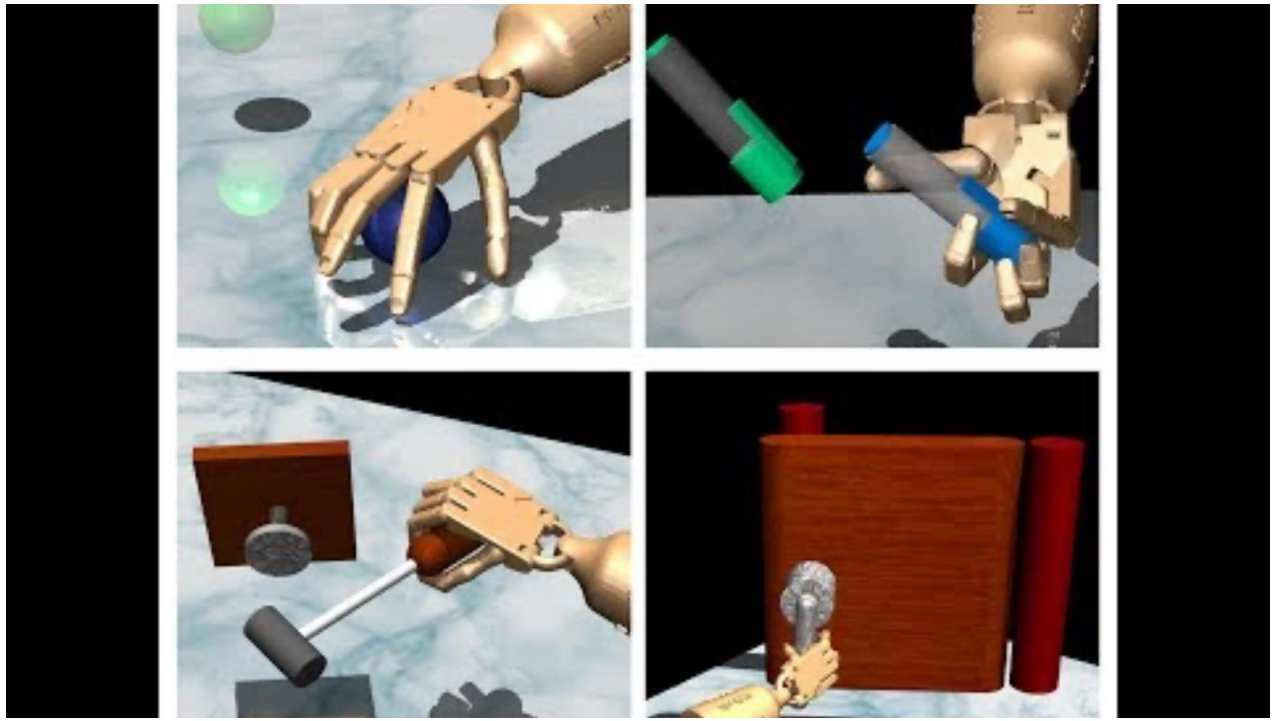
Most RL algorithms follow the same basic learning loop:



$$\tau = (x_0, u_0, \ldots, x_N, u_N)$$

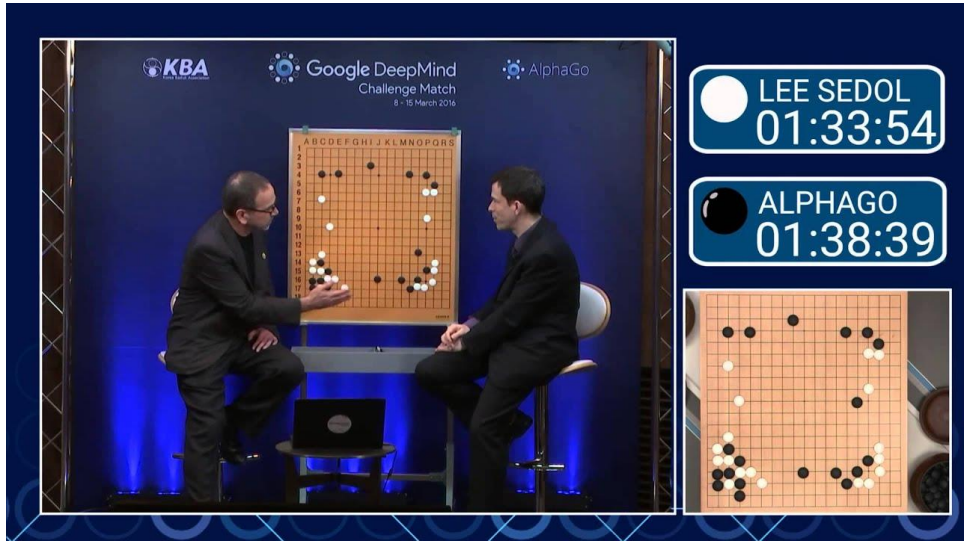
Why reinforcement learning?

- Only need to specify a reward function and the agent learns everything else!
  - This makes RL appealing for **complex physical tasks**



Why reinforcement learning?

- RL can discover new, unexpected solutions



Markov Decision Process

State: $x \in \mathcal{X}$

Action: $u \in \mathcal{U}$

Transition function / Dynamics: $T(x_t \mid x_{t-1}, u_{t-1}) = p(x_t \mid x_{t-1}, u_{t-1})$

Reward function: $r_t = R(x_t, u_t) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$

Discount factor: $\gamma \in (0,1)$

Typically represented as a tuple

$$\mathcal{M} = (\mathcal{X}, \mathcal{U}, T, R, \gamma)$$

Goal: choose a policy that maximizes cumulative (discounted) reward

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_p\left[\sum_{t \geq 0} \gamma^t R\big(x_t, \pi(x_t)\big)\right]$$

Some examples

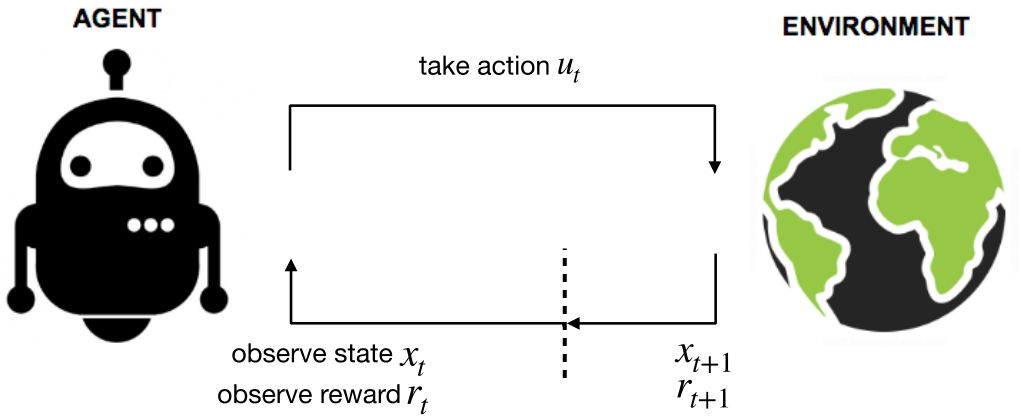
Chess

State:	$x \in \mathcal{X}$	*position of pieces*
Action:	$u \in \mathcal{U}$	*next move*
Dynamics	$p(x_t \mid x_{t-1}, u_{t-1})$	*given a move, defines the next configuration of the board (deterministic)*
Reward function:	$r_t = R(x_t, u_t)$	*-1/+1 for lost/won game*

Quadruped

State:	$x \in \mathcal{X}$	*robot's pose (e.g., foot positions, joint angles) and its position on the terrain*
Action:	$u \in \mathcal{U}$	*joint torque commands for each leg*
Dynamics	$p(x_t \mid x_{t-1}, u_{t-1})$	*the robot applies the chosen torque commands to its joints, deterministically resulting in the next pose and position based on the physics of the terrain*
Reward function:	$r_t = R(x_t, u_t)$	*+10 forward step / -5 falling / 0 standing still*

The goal of reinforcement learning



- The agent interacts with the environment to generate trajectories $\tau = (x_0, u_0, x_1, u_1, \ldots x_T)$
- We define the trajectory distribution

$$p(x_0, u_0, \ldots, x_T) = p(\tau) = p(x_0) \prod_{t=1}^{T} \pi(u_t \,|\, x_t) p(x_{t+1} \,|\, x_t, u_t)$$

- We can express the RL objective as an expectation under the trajectory distribution

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t \geq 0} \gamma^t R\left(x_t, u_t\right) \right]$$

Value functions

State-value function: "*the expected total reward if we start in that state and act accordingly to a particular policy*"

$$\mathrm{V}_{\pi}(x_t) = \mathbb{E}_p \left[\sum_{t' \geq t} \gamma^{t'} R\left(x_{t'}, \pi\left(x_{t'}\right)\right) \right]$$

Action-state value function: "*the expected total reward if we start in that state, take an action, and act accordingly to a particular policy*"

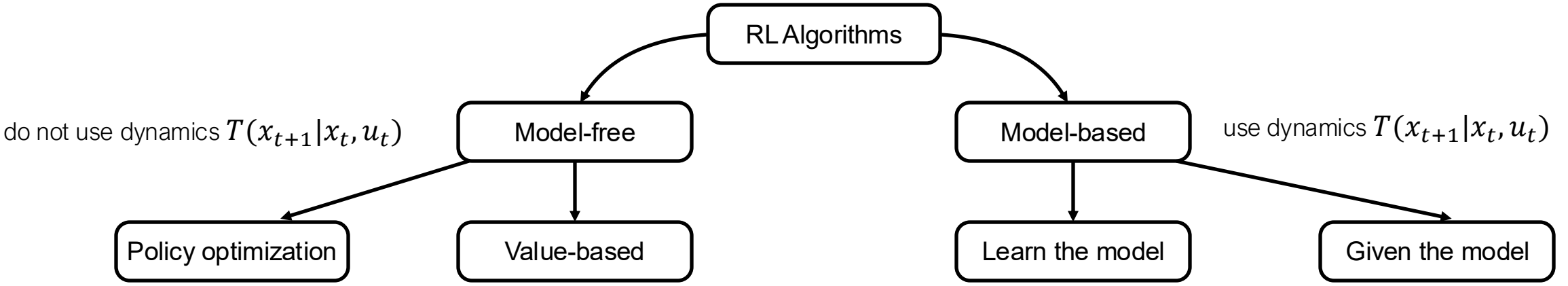
$$\mathrm{Q}_{\pi}(x_t, u_t) = \mathbb{E}_p \left[\sum_{t' \geq t} \gamma^{t'} R\left(x_{t'}, u_{t'}\right) \right]$$

Optimal state-value function: $V^*(x) = \max_{\pi} V_{\pi}(x)$

Optimal action-state value function: $Q^*(x, u) = \max_{\pi} Q_{\pi}(x, u)$

The majority of RL algorithms are defined via manipulations of these concepts

A taxonomy of RL



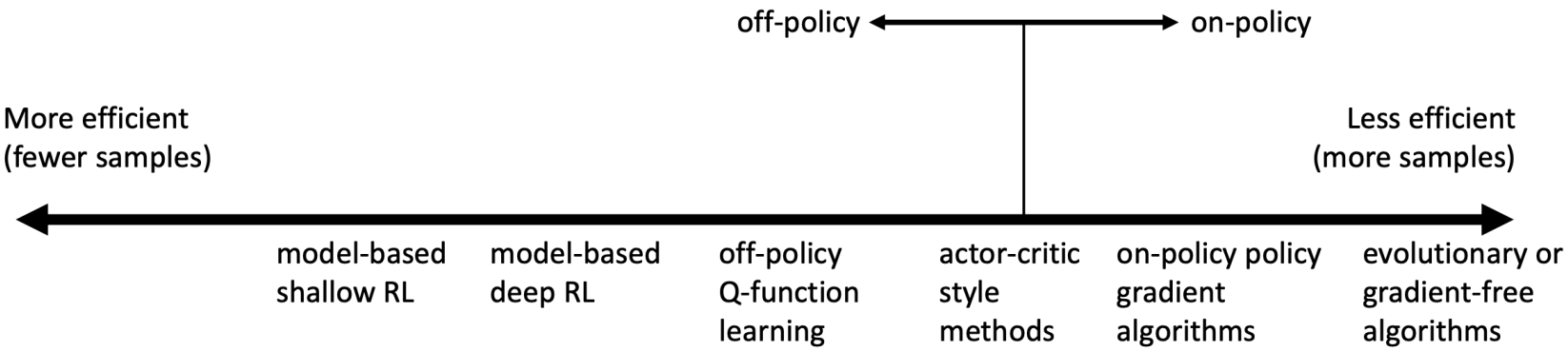
Why so many RL algorithms?

- Different tradeoffs:
  - Sample efficiency
  - Stability & easy of use
- Different assumptions:
  - Stochastic or deterministic
  - Continuous or discrete
  - Episodic or infinite horizon
- Different things are easy or hard in different settings:
  - Easier to represent the policy?
  - Easier to represent the model?

Comparison: sample efficiency

- Sample efficiency = how many samples do we need to get a good policy?

- Crucial question: is the algorithm *off policy?*
  - Off policy: able to improve the policy without generating new samples from the current policy
  - On policy: each time the policy is changed, even a little bit, we need to generate new samples



Why even bother using less efficient algorithms? Wall-clock time is not the same as efficiency!

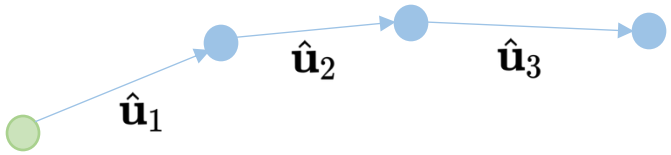
Comparison: stability and ease of use

- Does it converge?
- And if it does, to what?
- Does it *always* converge?

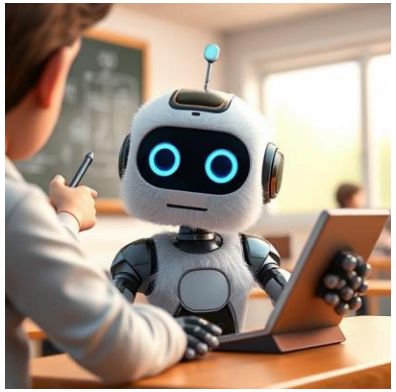
- Supervised learning: almost always gradient descent
- Reinforcement learning: often not gradient descent
  - Q-learning: fixed point iteration
  - Model-based RL: model estimator is not optimized for expected reward

Reinforcement Learning vs Imitation Learning

- IL: learn to imitate the expert's behavior
  - Albeit powerful, it is not suitable for learning from interaction
  - RL: "*learning without a teacher*"
- In RL there is a need to balance between exploration and exploitation
- Feedback may be delayed or sparse (i.e., credit assignment)
- Data is ***not*** IID

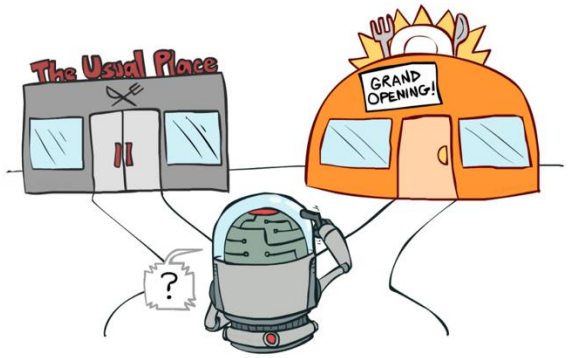


IL



vs

RL



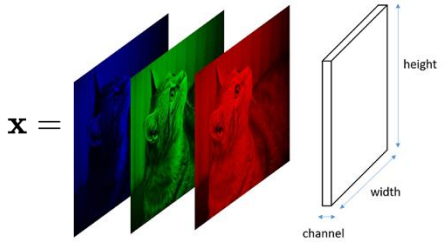
P(win | move)?

Outline

Primer on Supervised Learning

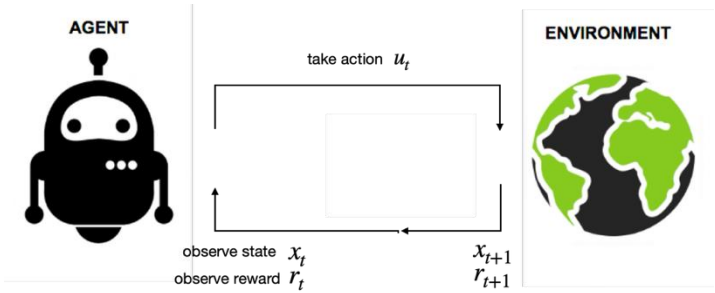
Imitation Learning

Reinforcement Learning



$f : \mathcal{X} \to \mathcal{Y}$ $\mathcal{Y} = \{0, 1, \ldots, C\}$

`e.g., [dog, cat, …, mouse]`



$$\tau = (x_0, u_0, \ldots, x_N, u_N)$$

Key learning goals:

- An overview of the fundamentals of IL and RL
- Identify the distinctions and similarities between these paradigms

Next class

- Imitation Learning (IL)