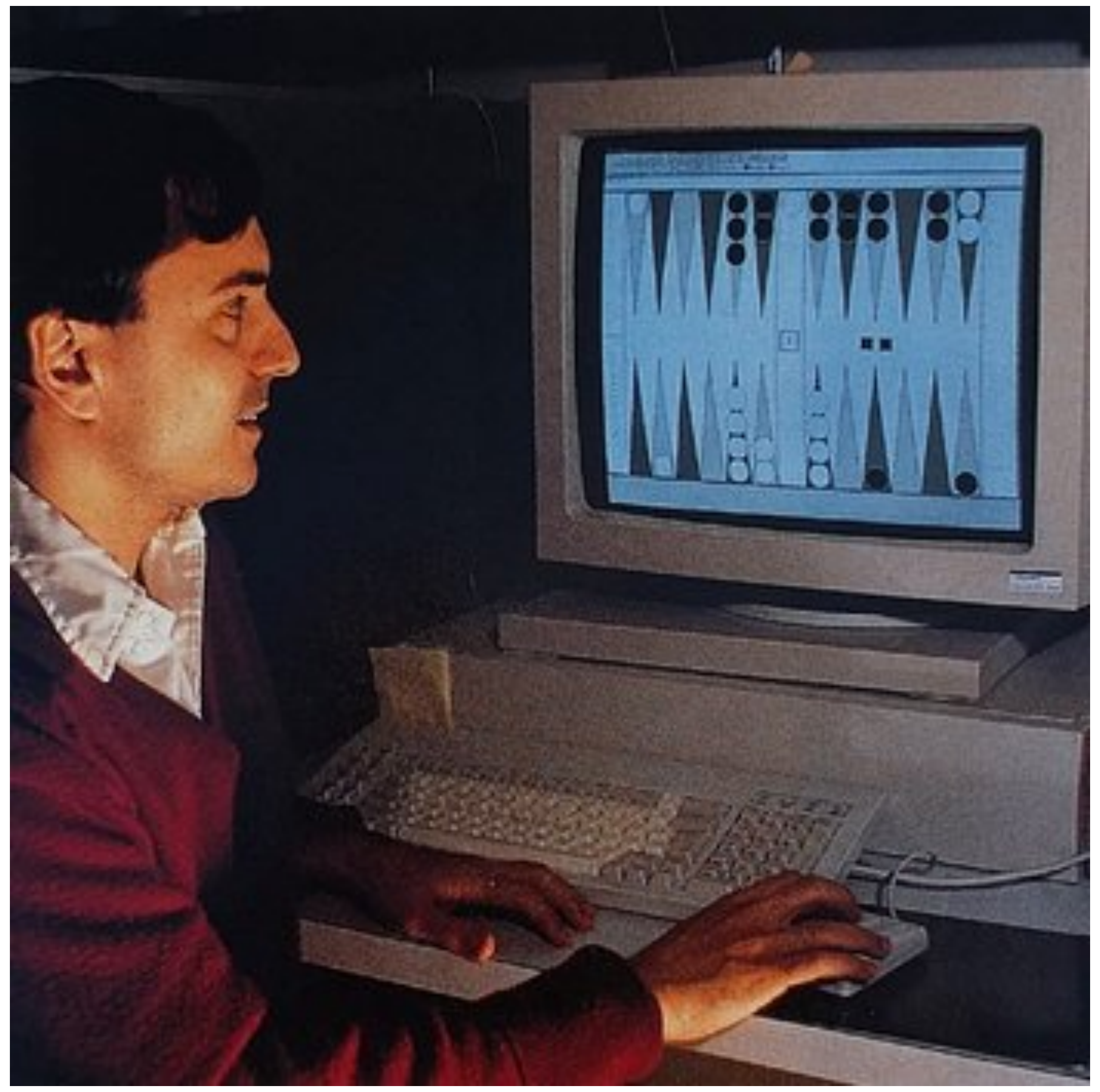
Introduction to Reinforcement Learning

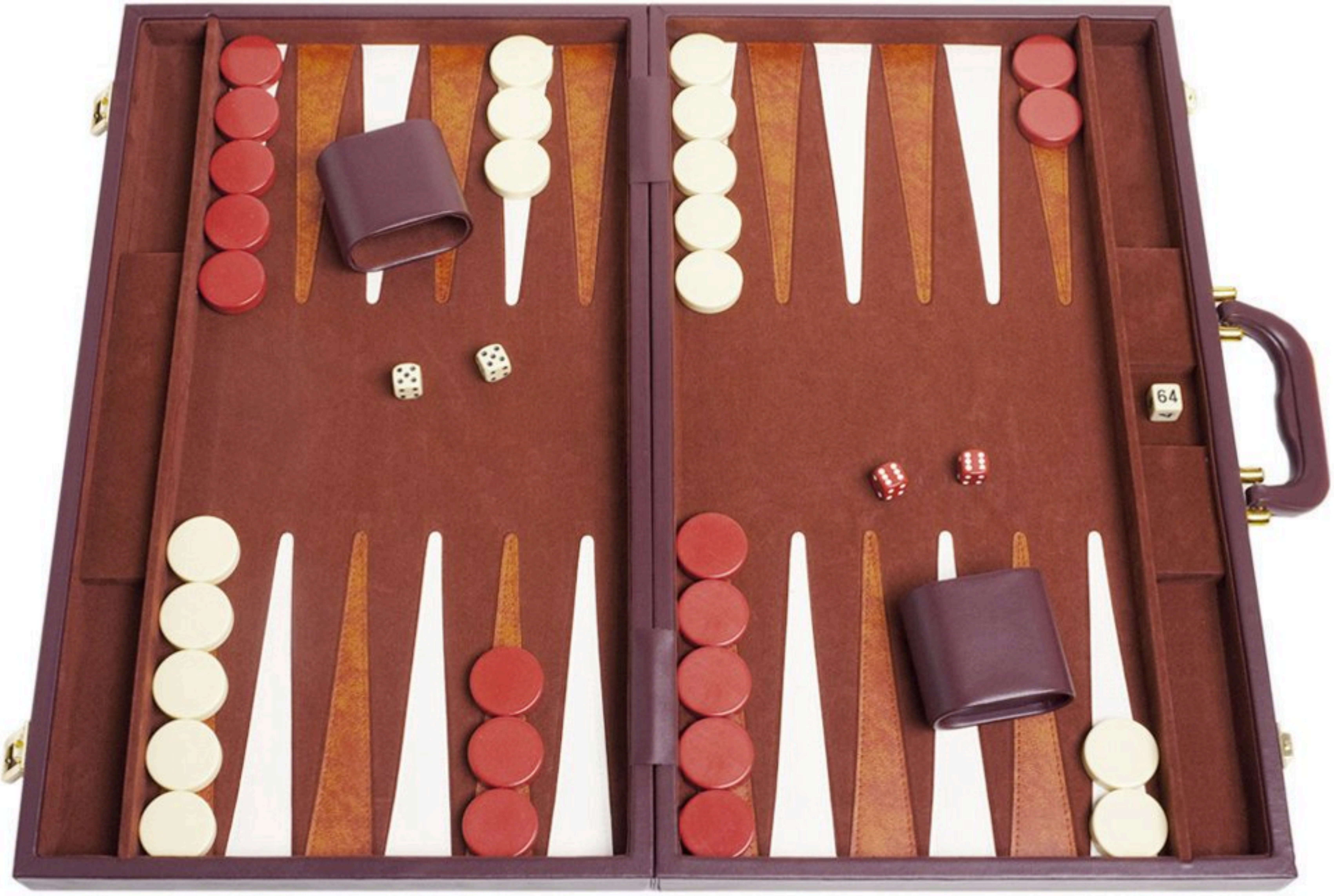
Matt Barnes

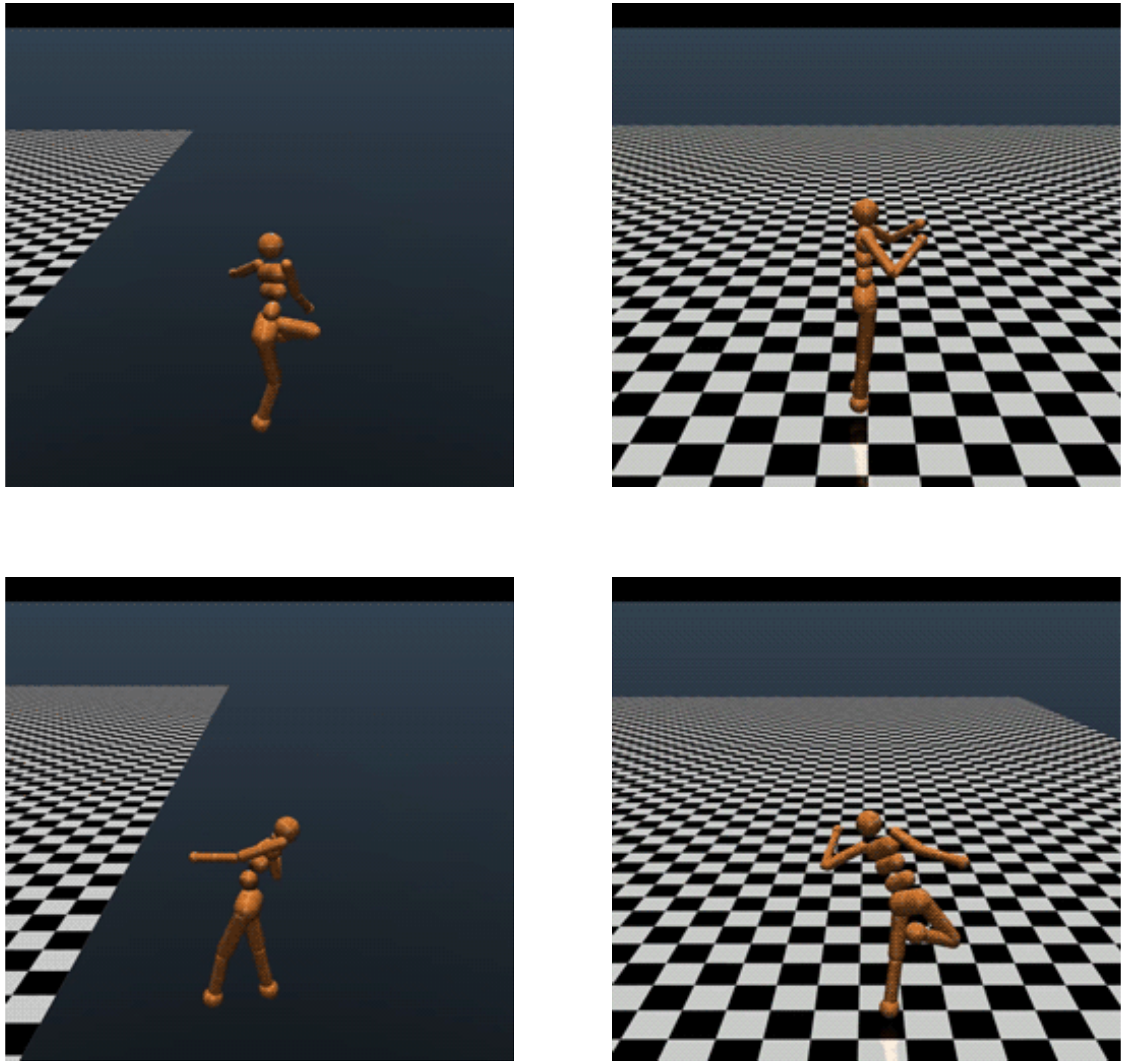
TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

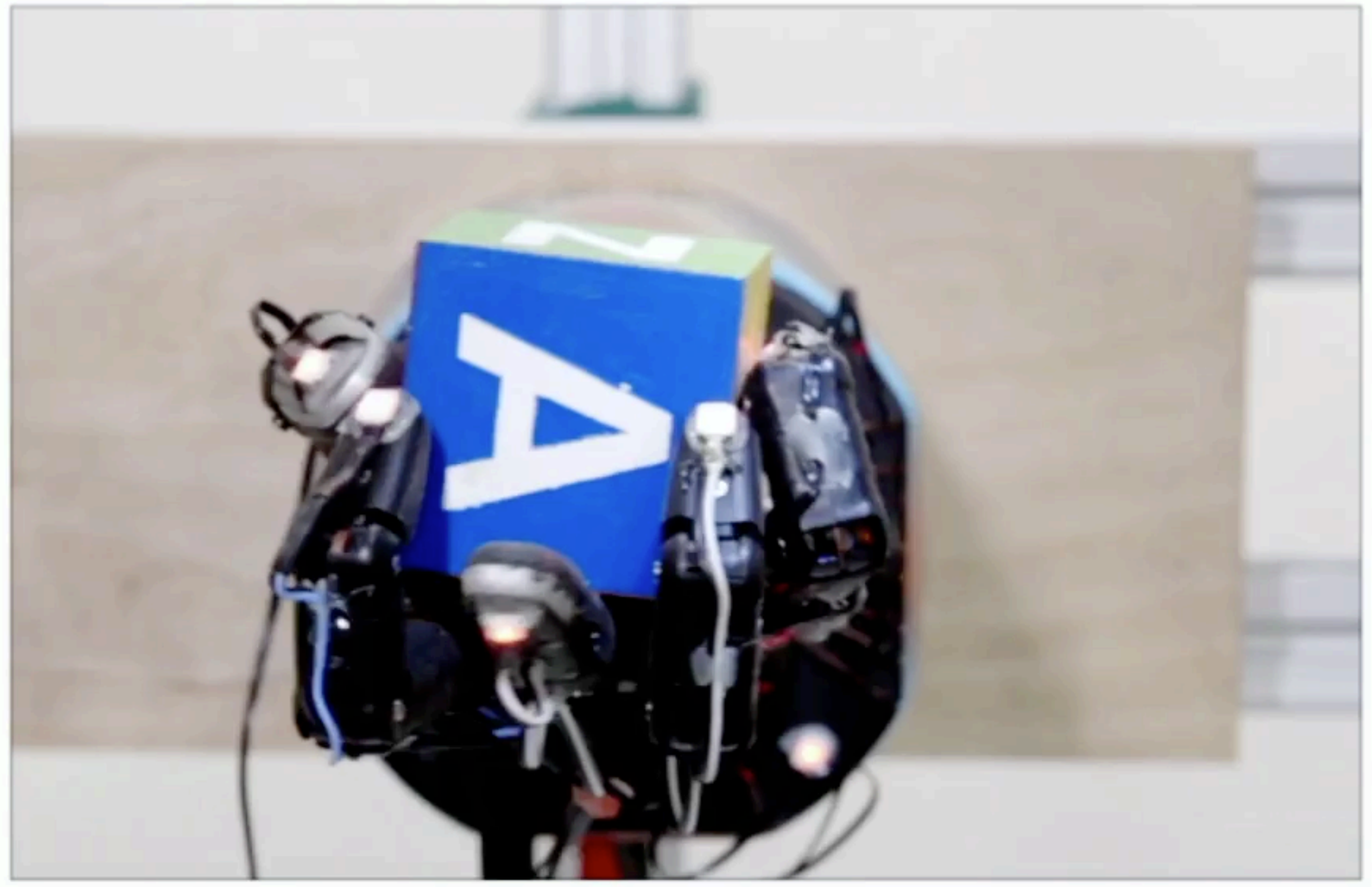
ALPHAGO



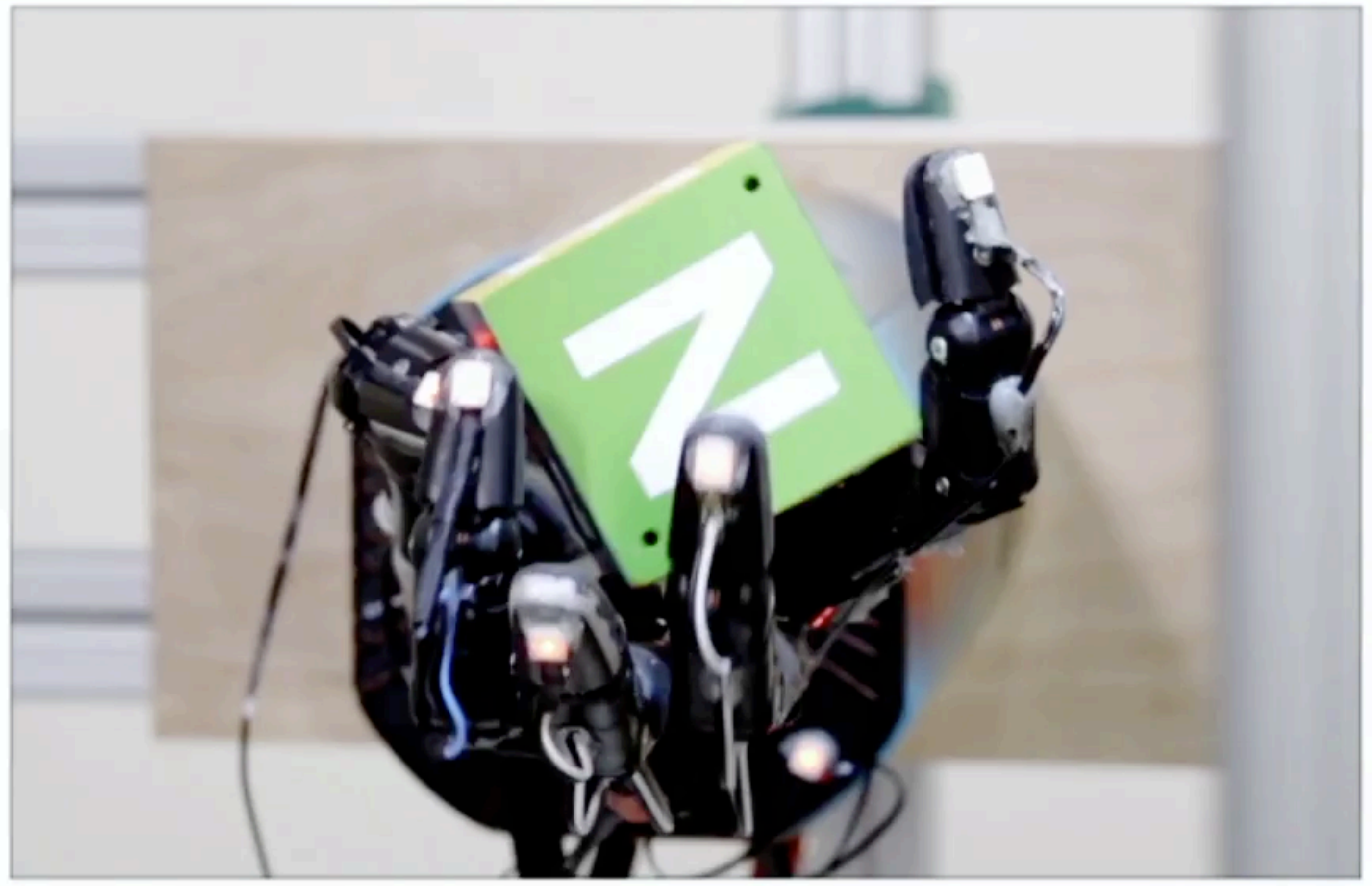
[Tesauro 1995]



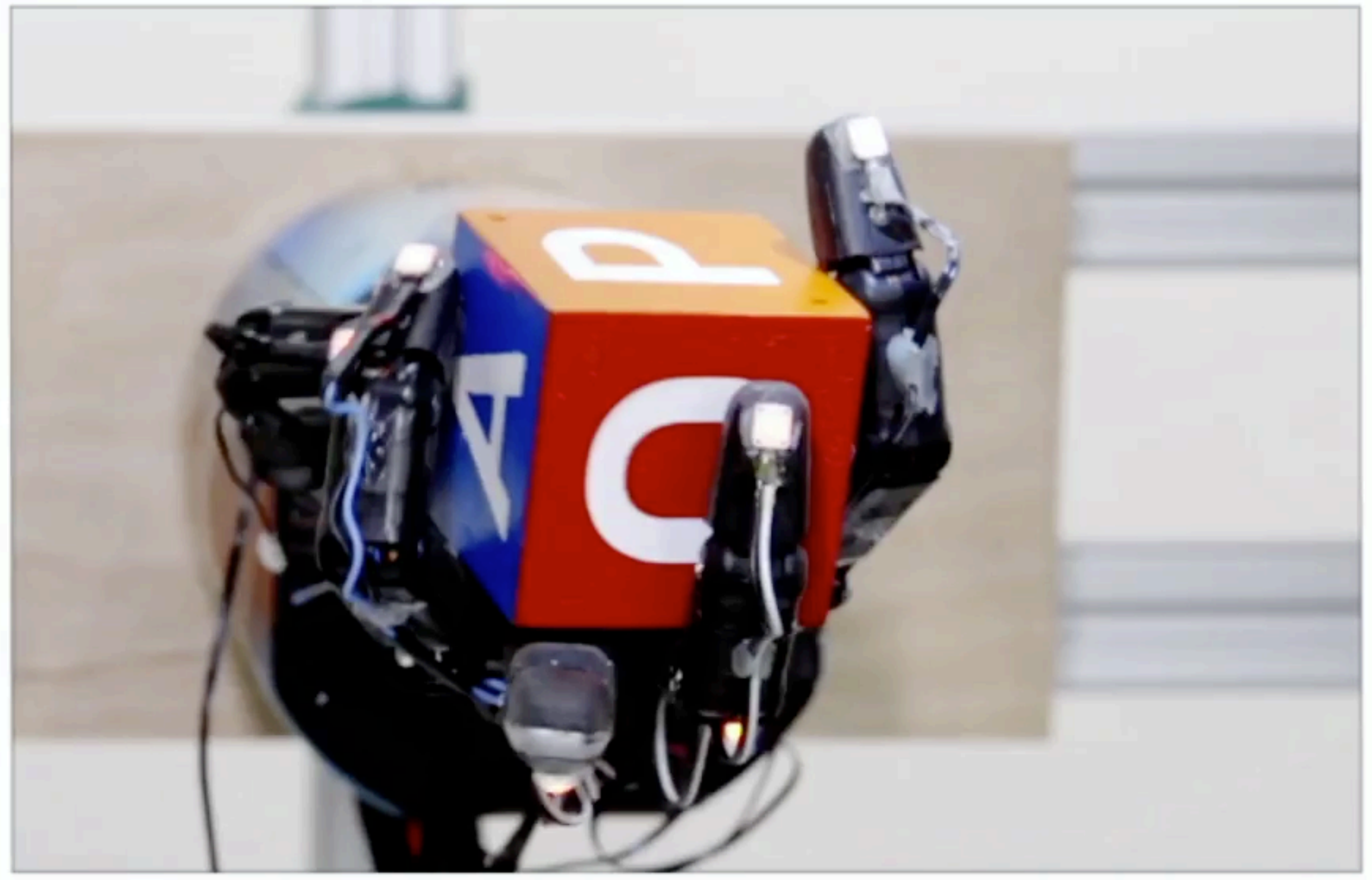




FINGER PIVOTING



SLIDING



FINGER GAITING

The elephant in the room: Should we use deep reinforcement learning to solve all our robotics problems?

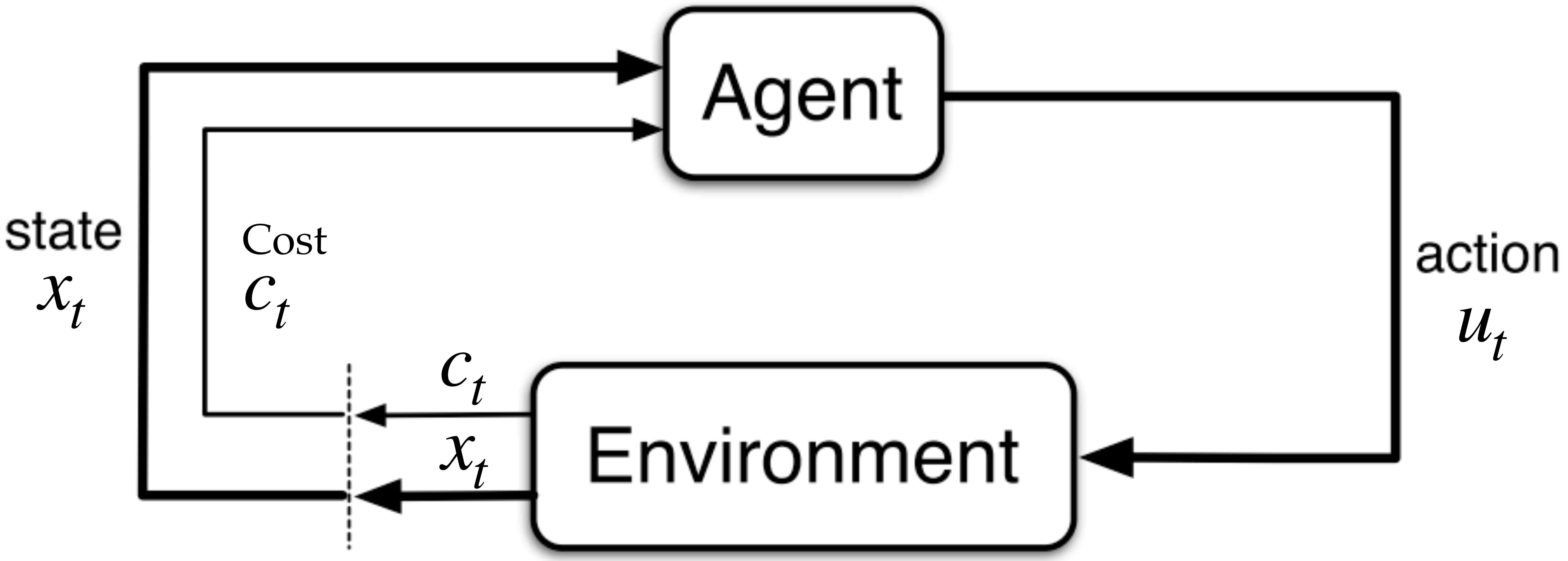
Today's Goals

1. Be able to define a Markov Decision Process is, and how it relates planning and reinforcement learning

2. Understand the high-level idea behind each of the 3 general approaches to reinforcement learning

3. How powerful function approximators like neural networks have contributed to recent successes in RL, and what their trade-offs are

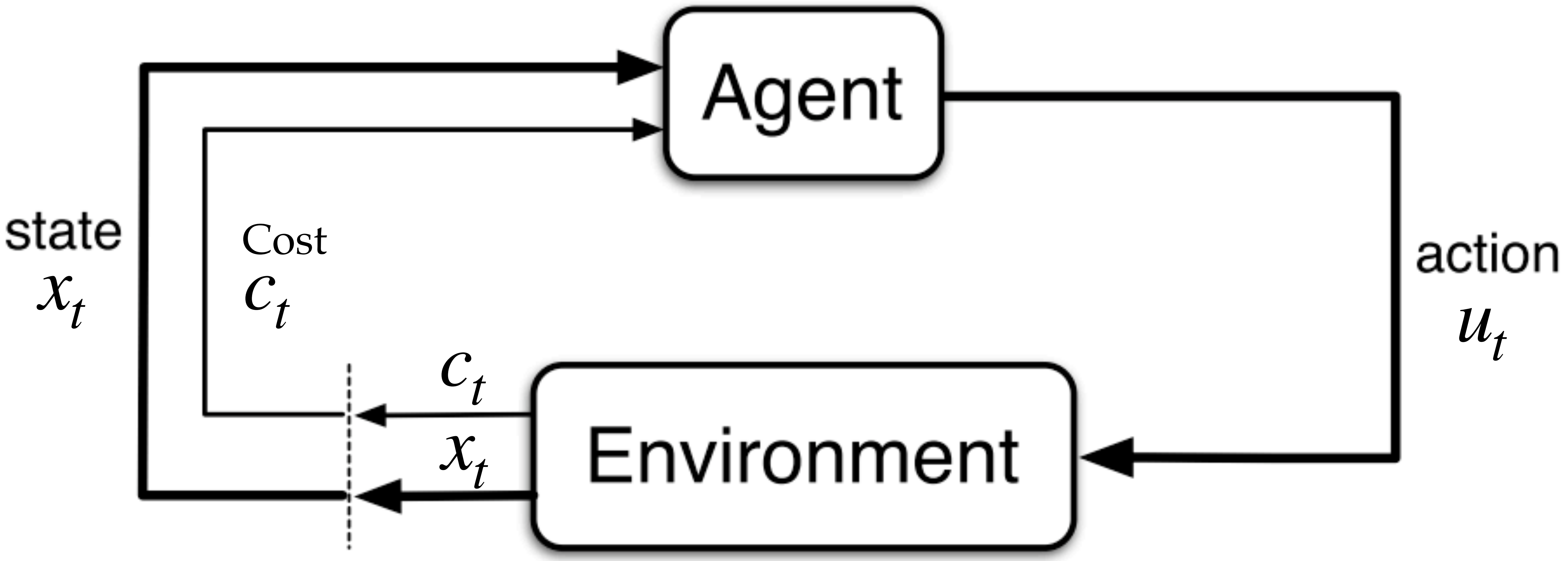
Markov Decision Processes



- Agent and environment interact at discrete time steps: $t = 0, 1, 2, \ldots$.
- Agent observes state x_t at time t
- Agent takes action $u_t \sim \pi(x_t)$
- Environment produces cost $c_t \sim c(s_t, u_t)$ and next state $s_{t+1} \sim T(s_t, u_t)$

The transition function / the dynamics

Markov Decision Processes

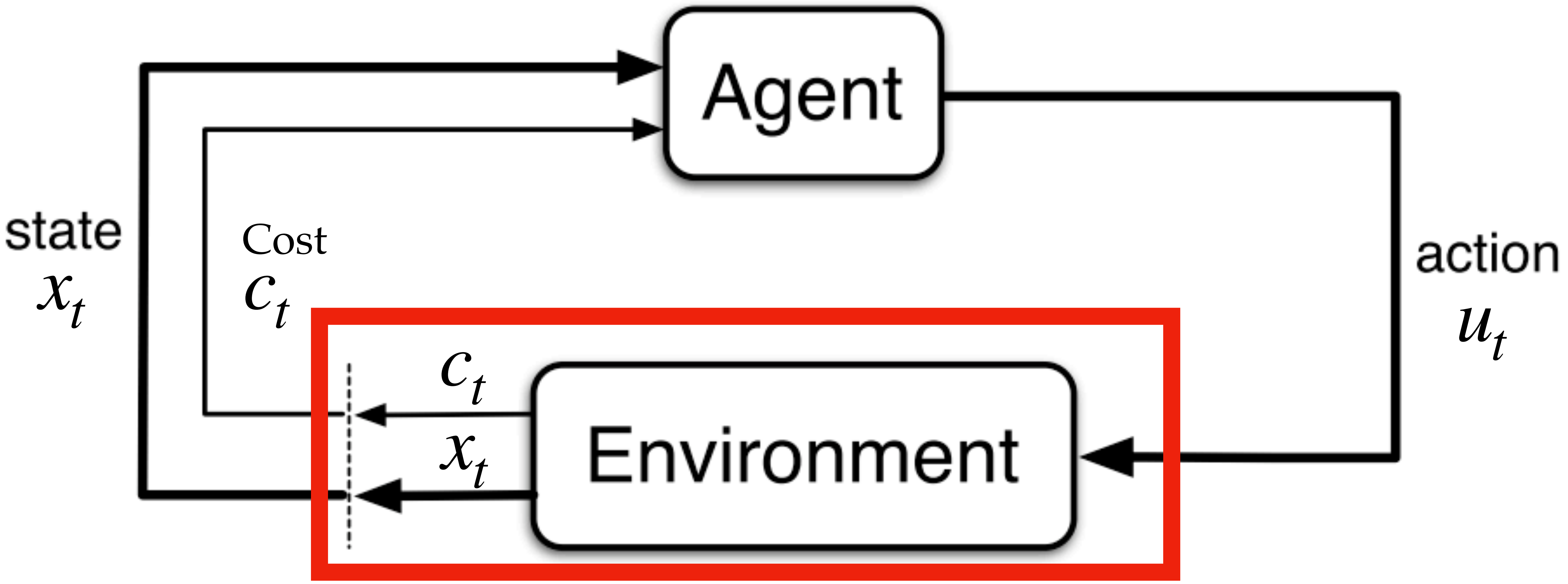


Our goal is to learn a policy which minimizes the **cumulative cost**

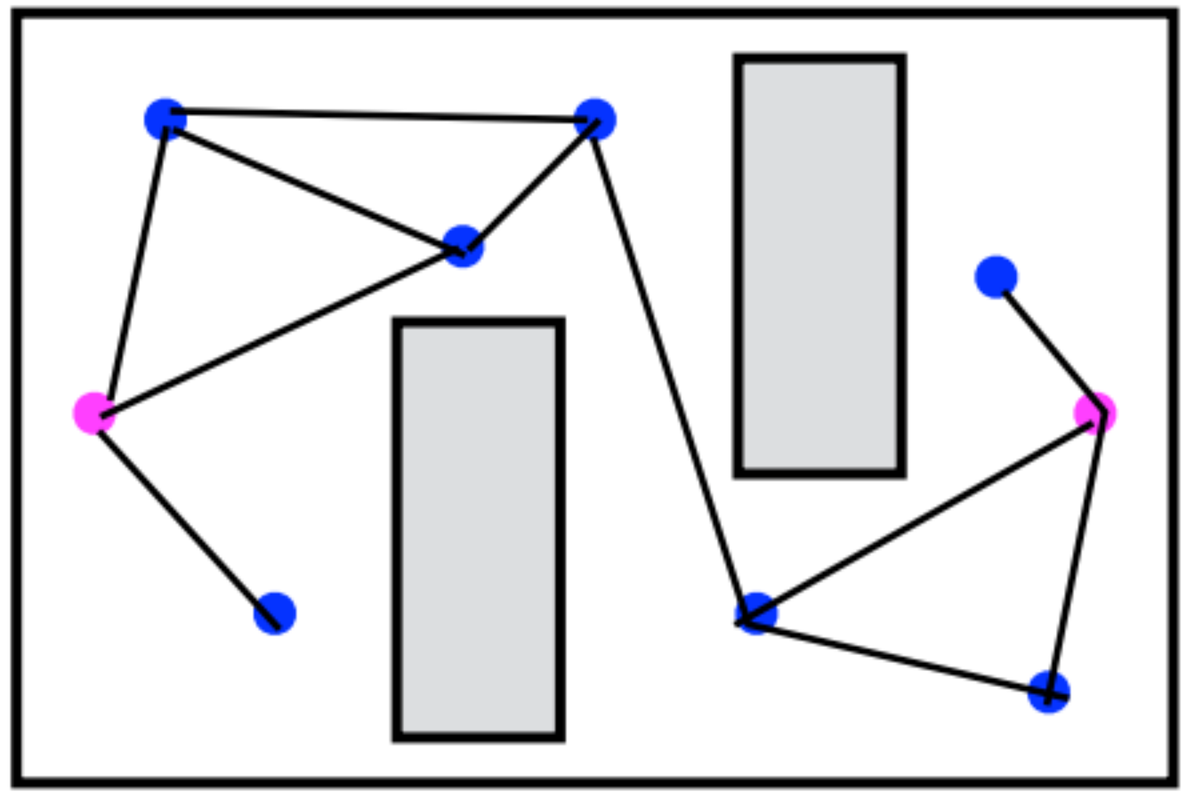
The time horizon

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} c(x_t, \pi(x_t))\right]$$

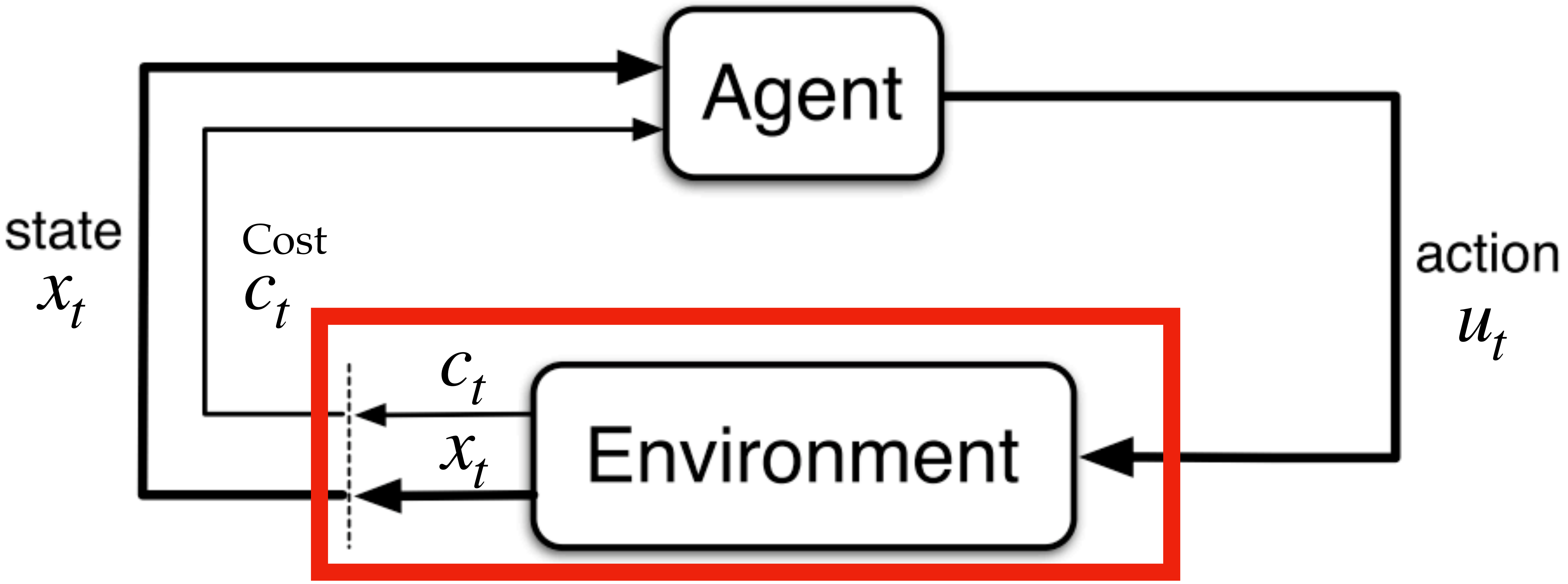
Planning as an MDP



Known (deterministic) environment



Planning as an MDP

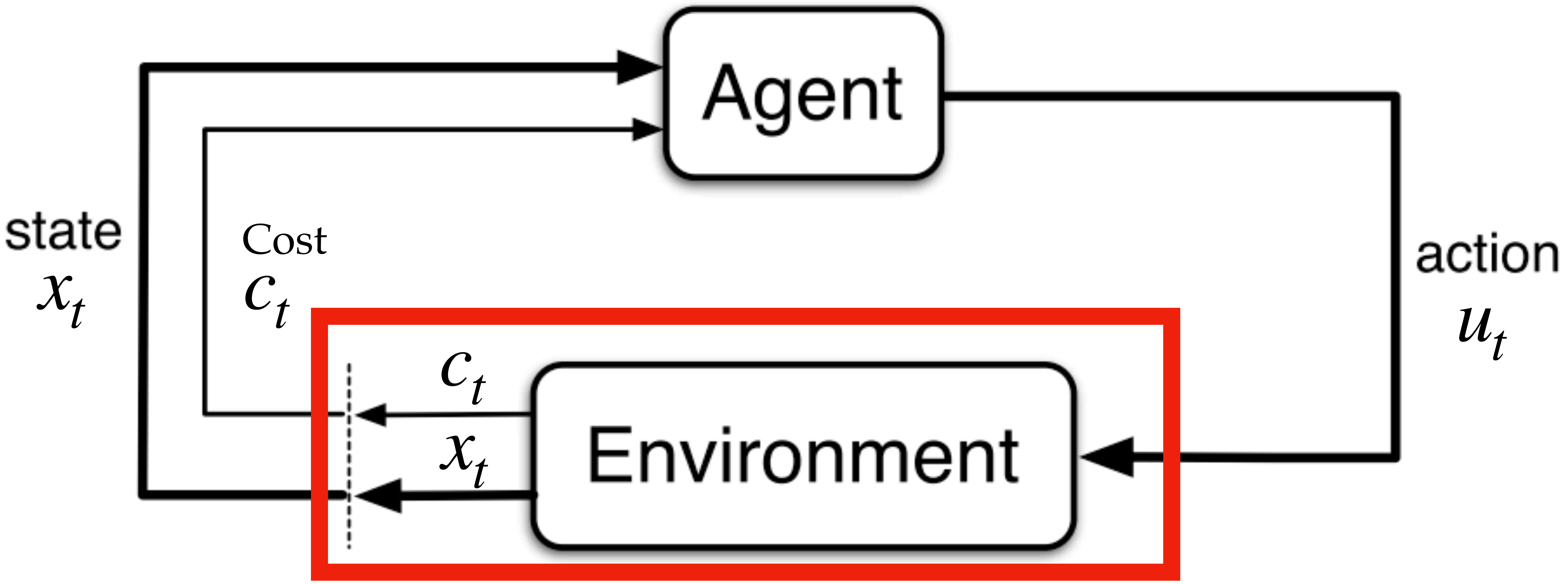


Known (deterministic) environment

$$x_{t+1} = Ax_t + Bu_t$$

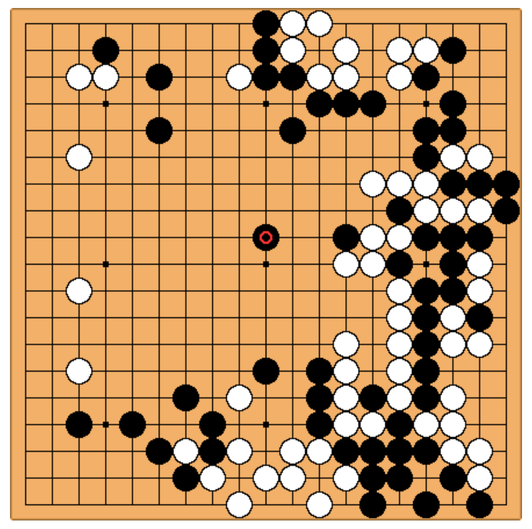
$$c(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

Reinforcement Learning as an MDP



Unknown (stochastic) environment

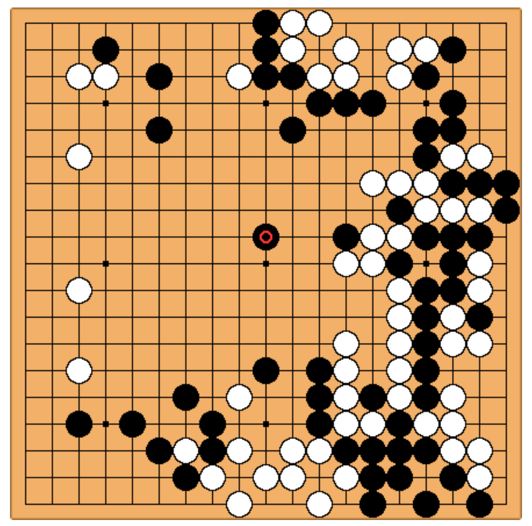
We must learn about the environment by interacting with it.



x_t

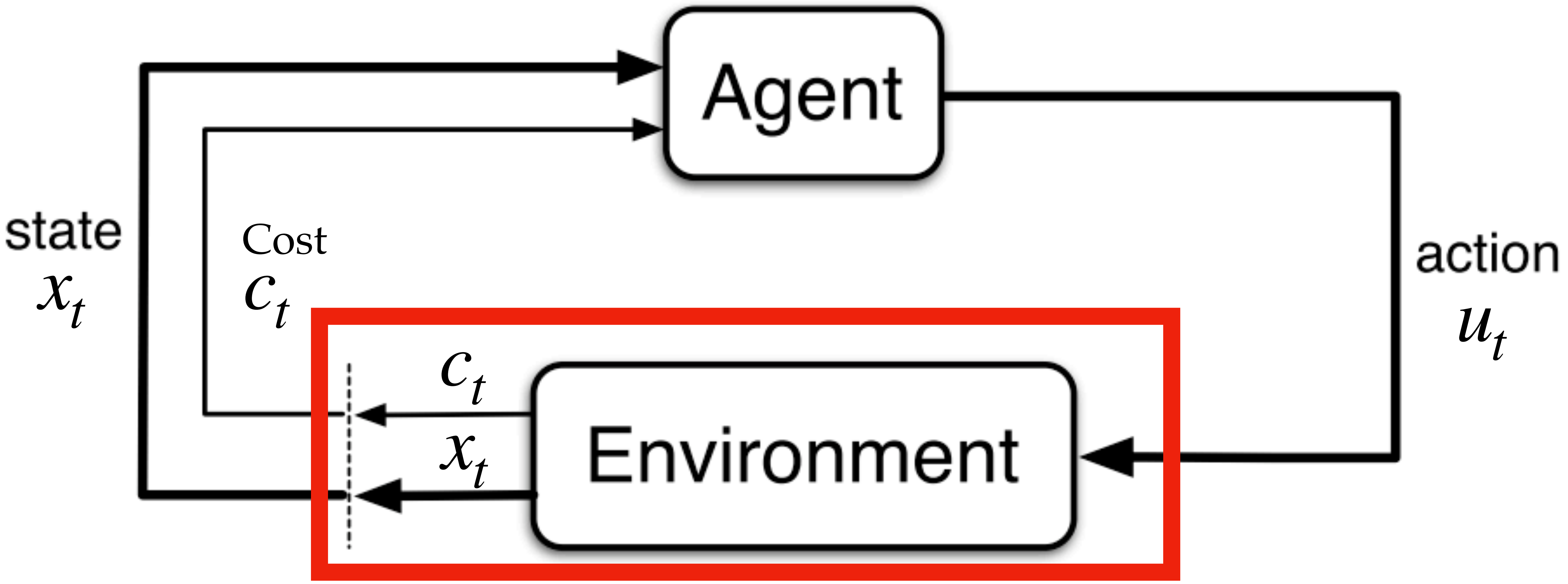
u_t + opponent move

Unknown & stochastic



x_{t+1}

Reinforcement Learning as an MDP



Unknown (stochastic) environment

Our goal is to learn a policy which minimizes the cumulative discounted cost **in the fewest interactions with the environment**

This is generally referred to as the **sample complexity** of the algorithm

We may also care about the **computational complexity** and **memory complexity**

"Markov" = Complete state

$$x_{t+1} = Ax_t + Bu_t$$

$$c(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

x_t tells us everything about the state of the system

Knowing $x_1, \ldots, x_{t-1}$ provides no additional information in determining x_{t+1} or $c(x_t, u_t)$

How can we solve RL?

Approach 1: Model-based

- Step 1: *Learn* a model of the environment using data
- Step 2: *Plan* using this model
- The "system identification" or "certainty equivalence" method

Model-free approaches

- Approach 2: Learn a value function using *approximate dynamic programming*
- Approach 3: Directly learn a policy using *policy gradient*

How can we solve RL?

Model-based approaches

- Intuitive and understandable representation

- Easier to incorporate prior information

Model-free approaches

- Promises of better theoretical sample complexity

Linear model-based RL

Step 1: Learn a model of the environment using data

- Dataset: $x_1, u_1, x_2, u_2, x_3, u_3, \ldots..$
- Perform linear regression to learn $\hat{A}, \hat{B}$

$$x_{t+1} = Ax_t + Bu_t$$

Step 2: Compute the optimal LQR controller $u_{t+1} = \hat{K}x_t$

$$\hat{K} = -(R + \hat{B}^T\hat{V}\hat{B})^{-1}\hat{B}^T\hat{V}\hat{A}$$

Model-free RL

Approach 2: Approximate dynamic programming

- **Q-Learning**
- Deep Q-Learning [Mnih 2015]

Approach 3: Policy gradient

- **REINFORCE [Williams 1992]**
- TRPO [Schulman 2015]
- PPO [Schulman 2017]
- DPG [Silver 2014]
- Actor-critic

Approximate dynamic programming methods learn a *value function*

Value function $Q_\pi(x, u)$ is the expected future (discounted) cost if we:

- Start from state x
- Take action u
- Then follow policy π

$$Q_\pi(x, u) = \mathbb{E}_{x' \sim T(x,u)} \left[c(x, u) + \gamma Q_\pi(x', \pi(x')) \right]$$

Expectation operator: Weighted average over possible next states s'

The one-step cost

The discount factor

The discounted future cost

Note this is a recursive formula — compute via **dynamic programming**

Approximate dynamic programming methods learn a *value function*

Bellman optimal value function, the value function of the *optimal policy*

$$Q^*(x, u) = \mathbb{E}_{x' \sim T(x,u)} \left[c(x, u) + \textbf{argmin}_{u'} Q^*(x', u') \right]$$

Always take the best action

The corresponding optimal policy

$$u_t = \pi^*(x_t) = \textbf{argmin}_{u'} Q^*(x_t, u')$$

Policy Gradient methods directly learn a policy

Step 1: Parameterize our policy π_θ, where θ are the parameters

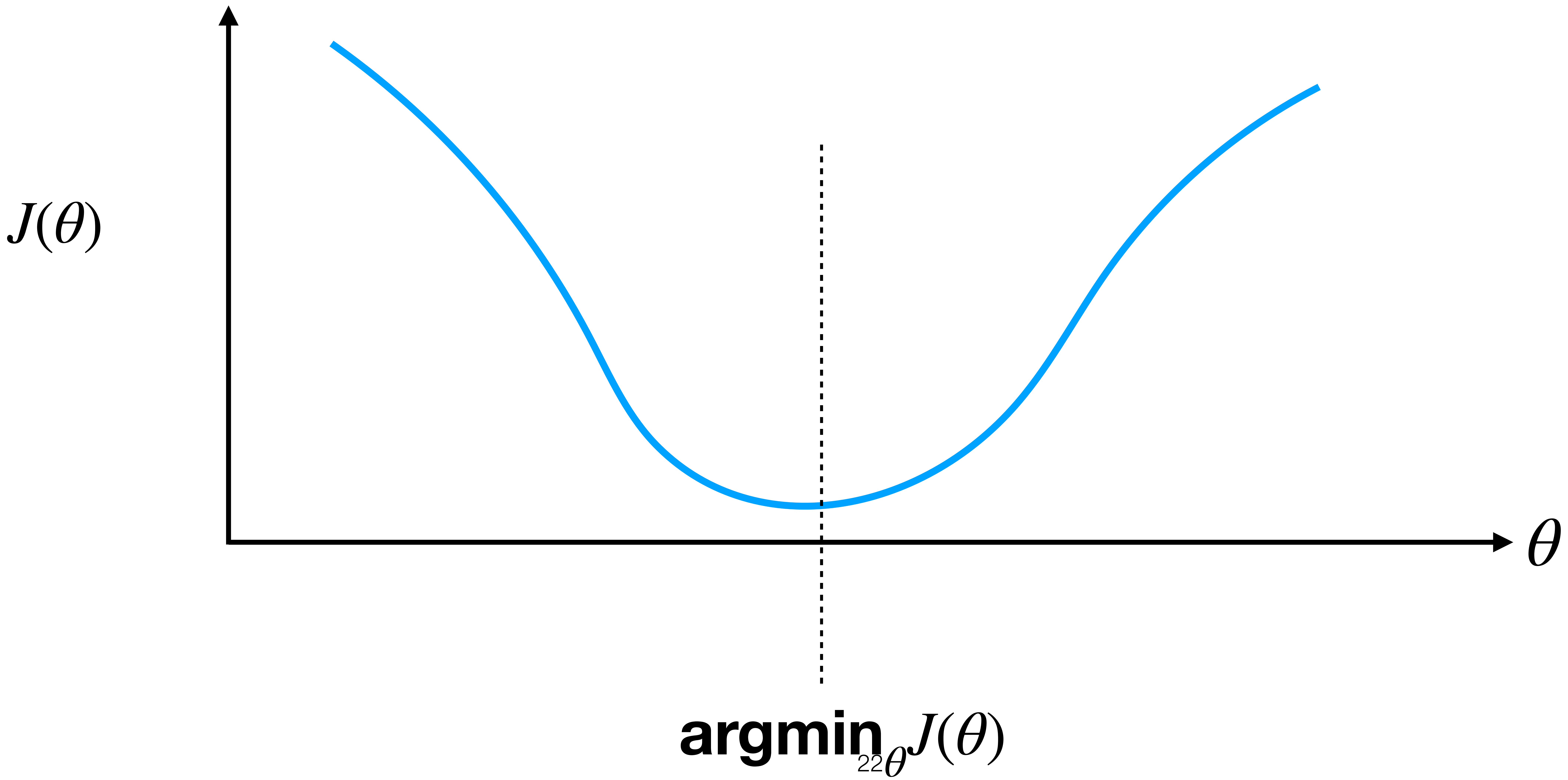
Ex. In LQR, we consider linear controllers $\pi_\theta(s) = \theta s$

Let $J(\theta) = Q_{\pi_\theta}(s_0, \pi_\theta(s_0))$ be the cumulative cost of this policy starting from the initial state

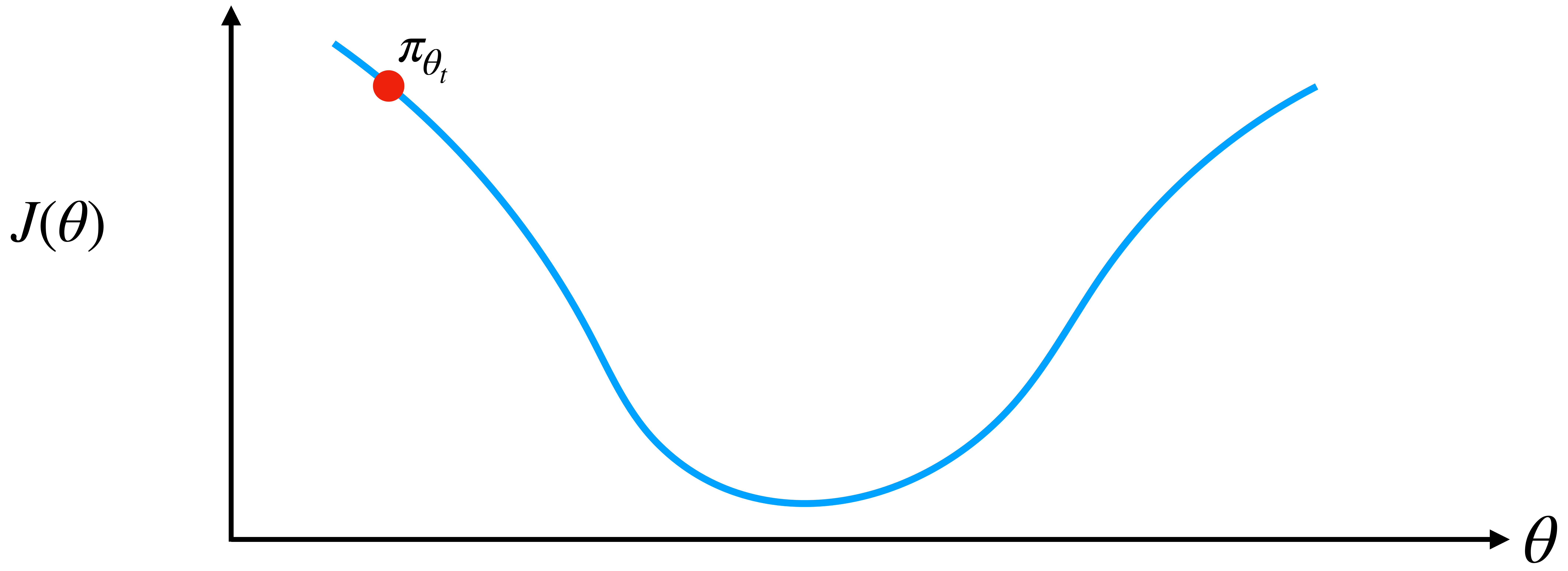
Step 2: Optimize!

$$\textbf{argmin}_\theta J(\theta)$$

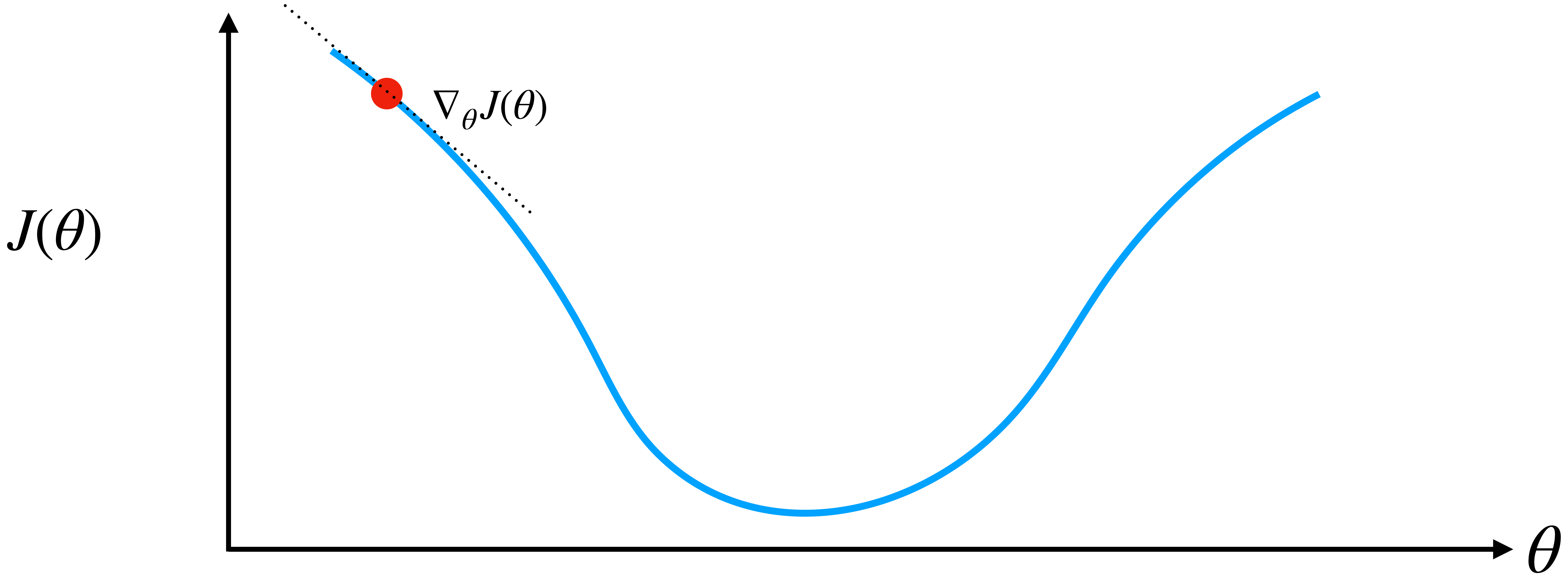
A brief overview of optimization



A brief overview of optimization



A brief overview of optimization

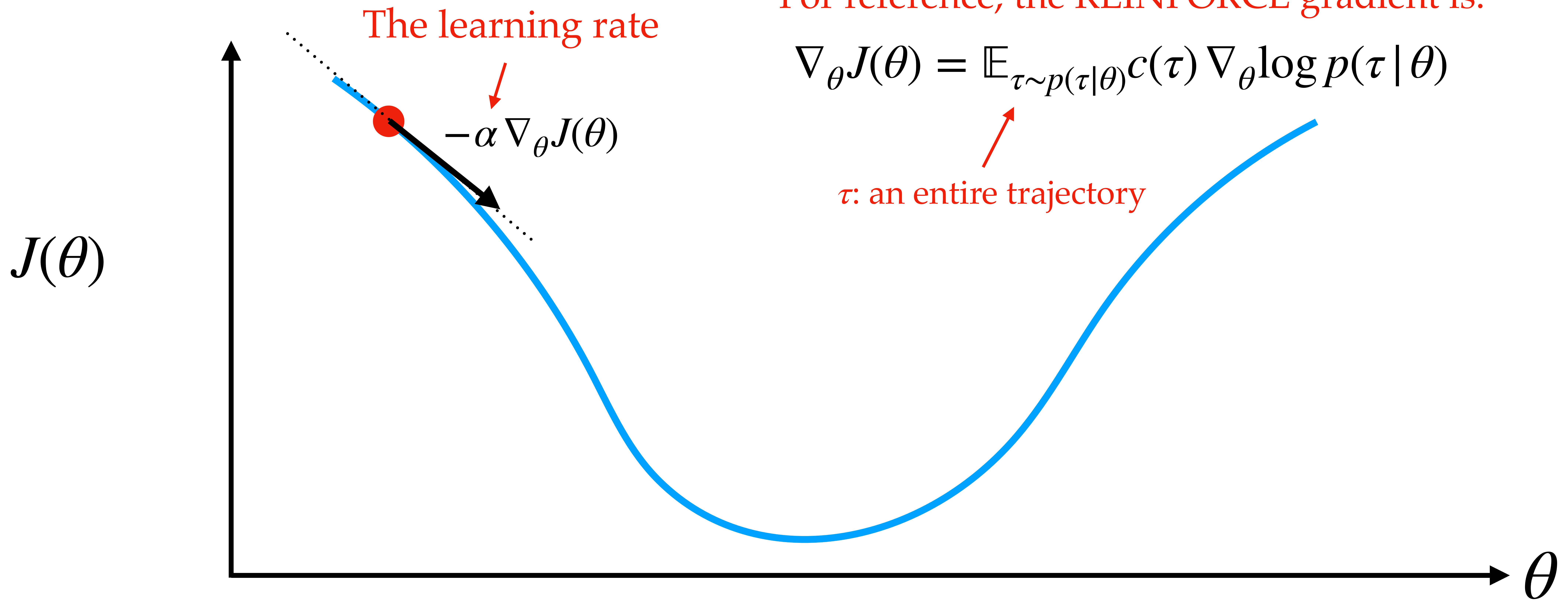


A brief overview of optimization

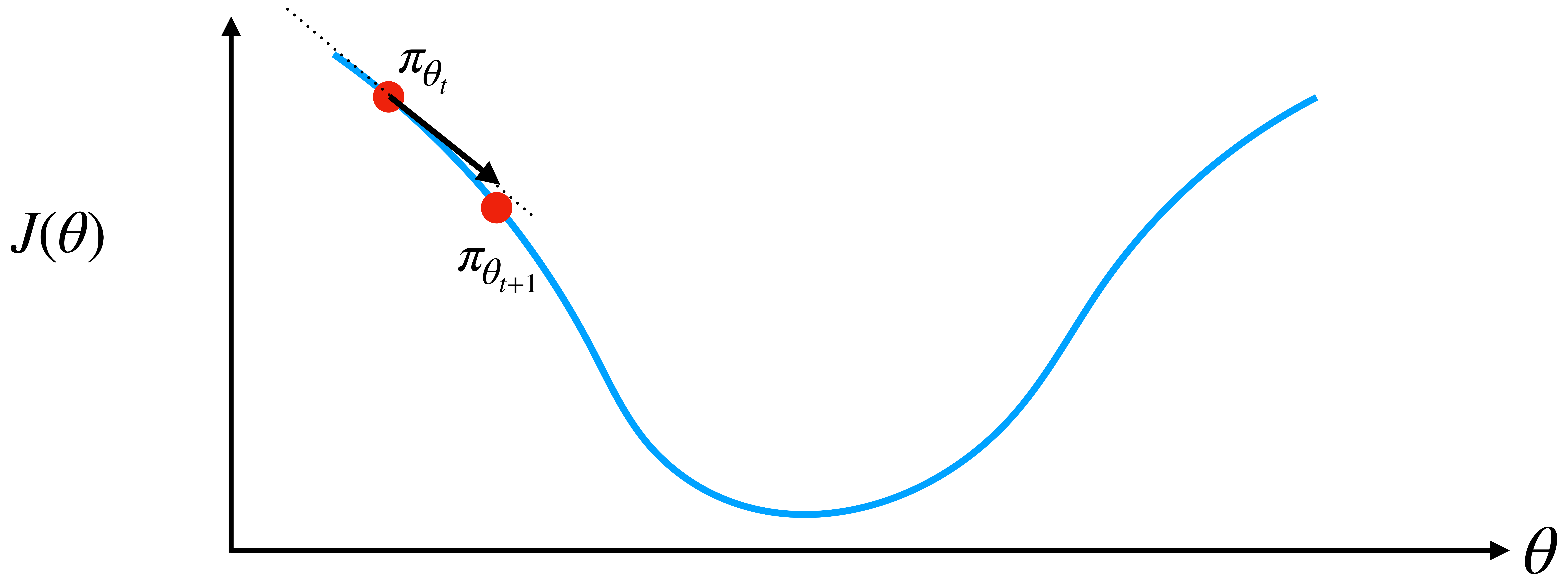
For reference, the REINFORCE gradient is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} c(\tau)\, \nabla_\theta \log p(\tau\,|\,\theta)$$

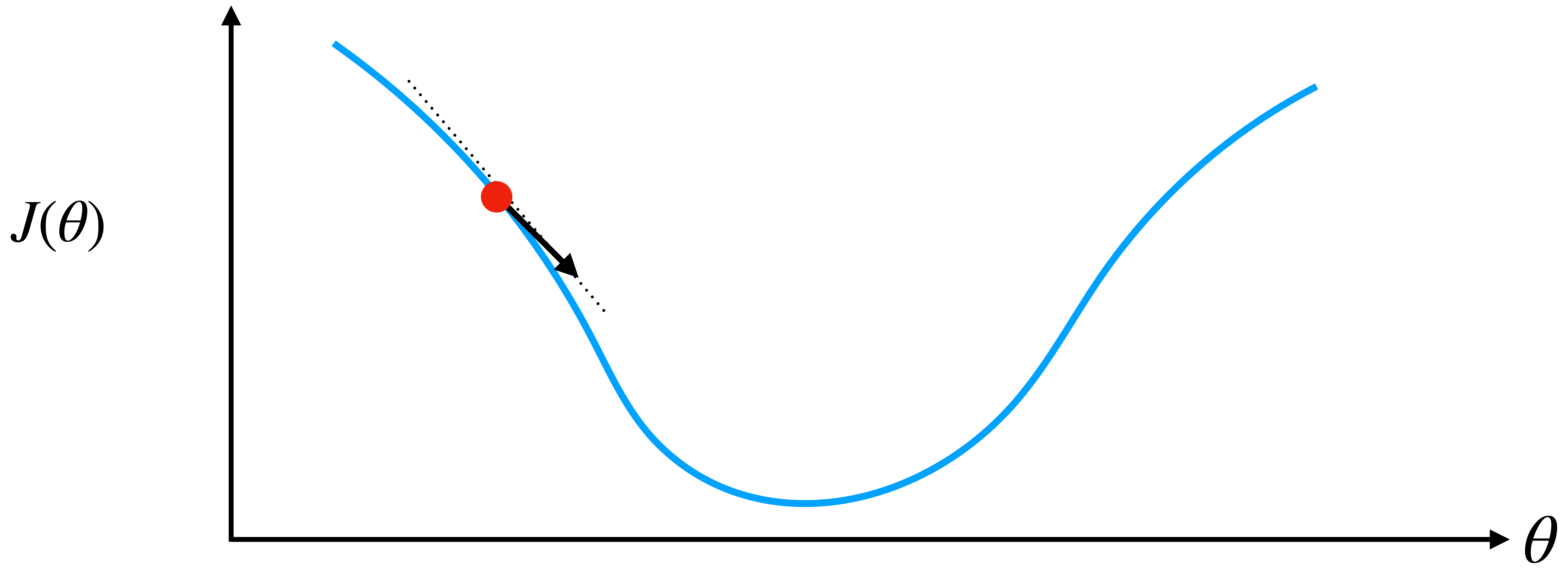
τ: an entire trajectory



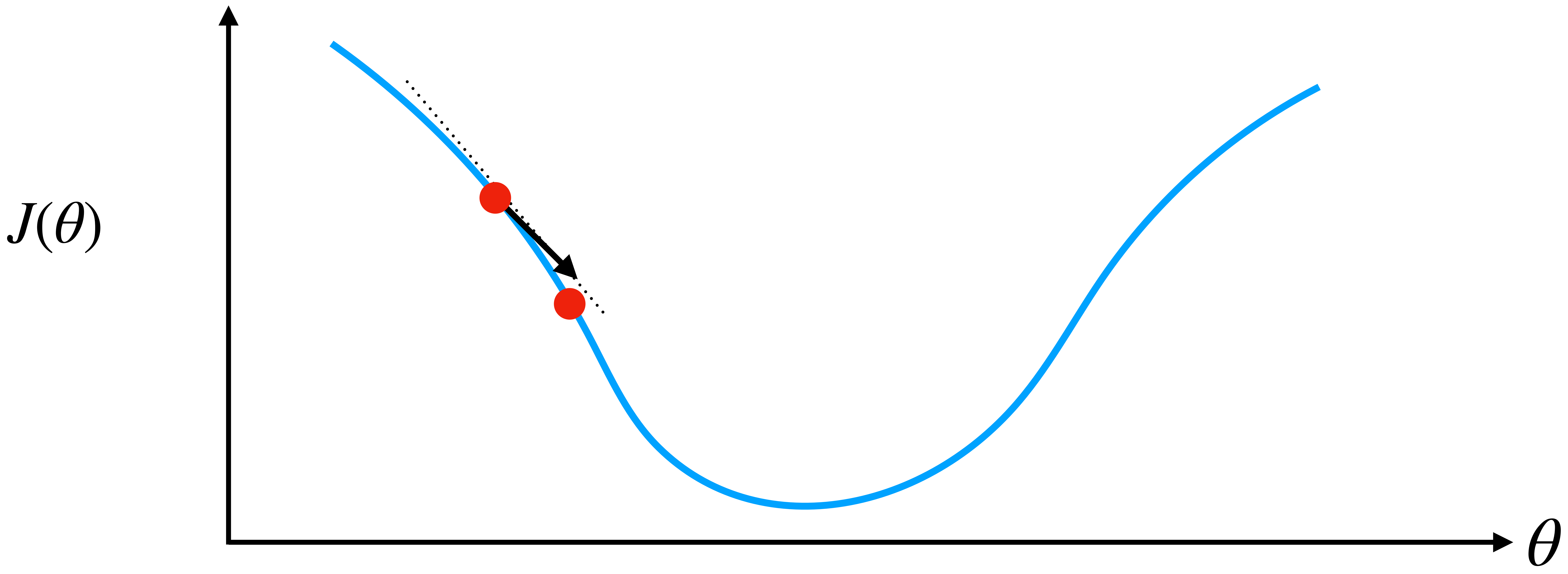
A brief overview of optimization



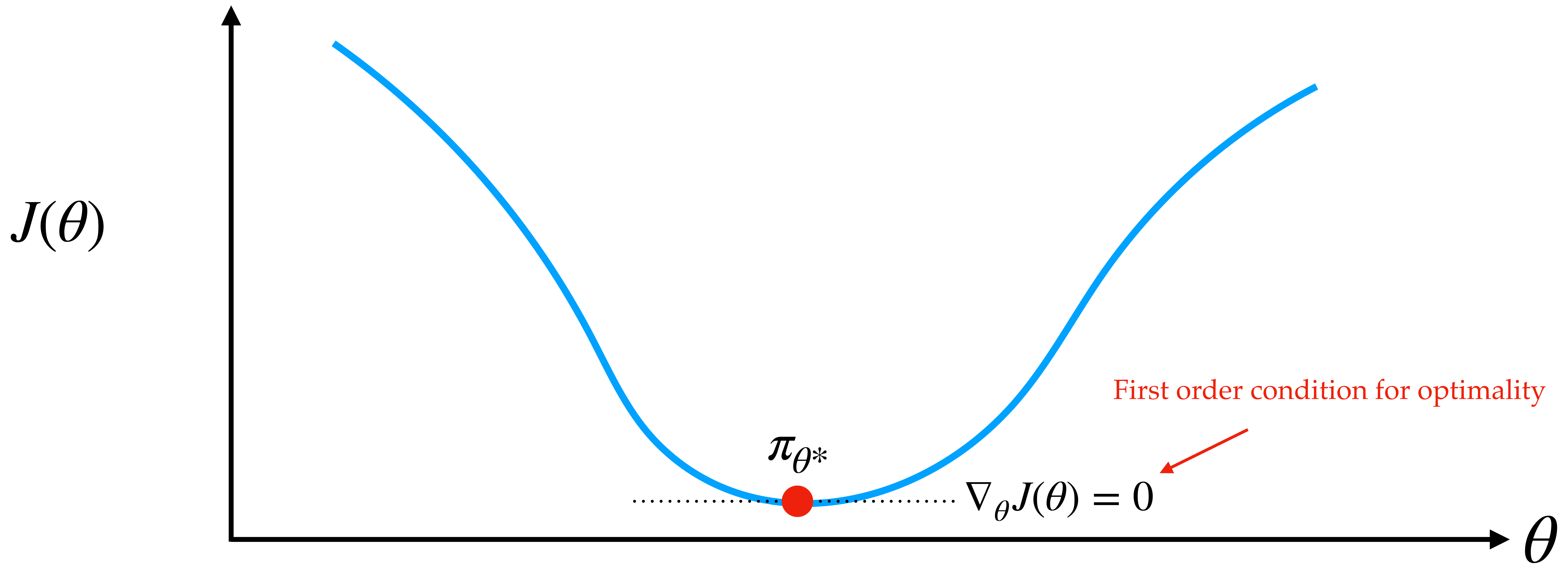
A brief overview of optimization



A brief overview of optimization



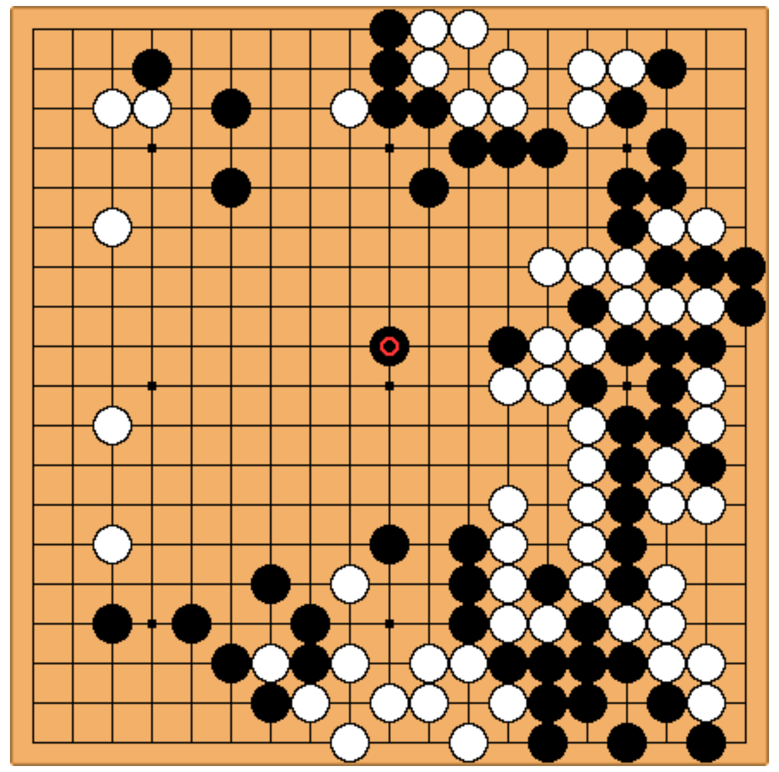
A brief overview of optimization

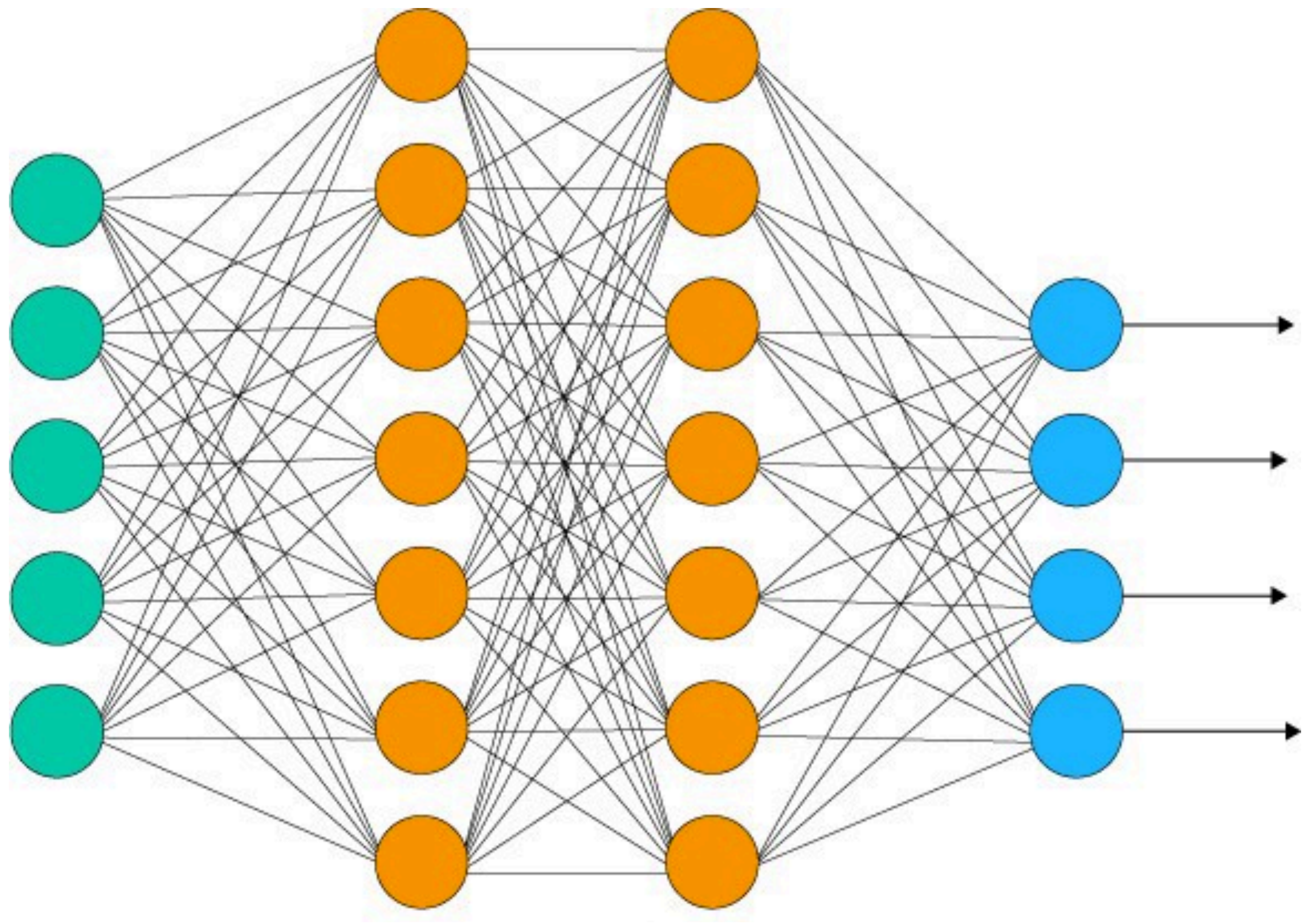


$\textbf{argmin}_\theta J(\theta)$

The power of function approximators

- Approximate dynamic programming: Q_θ
- Policy gradient: π_θ
- Instead of linear or tabular functions, we can use complex, nonlinear function approximators like neural networks!

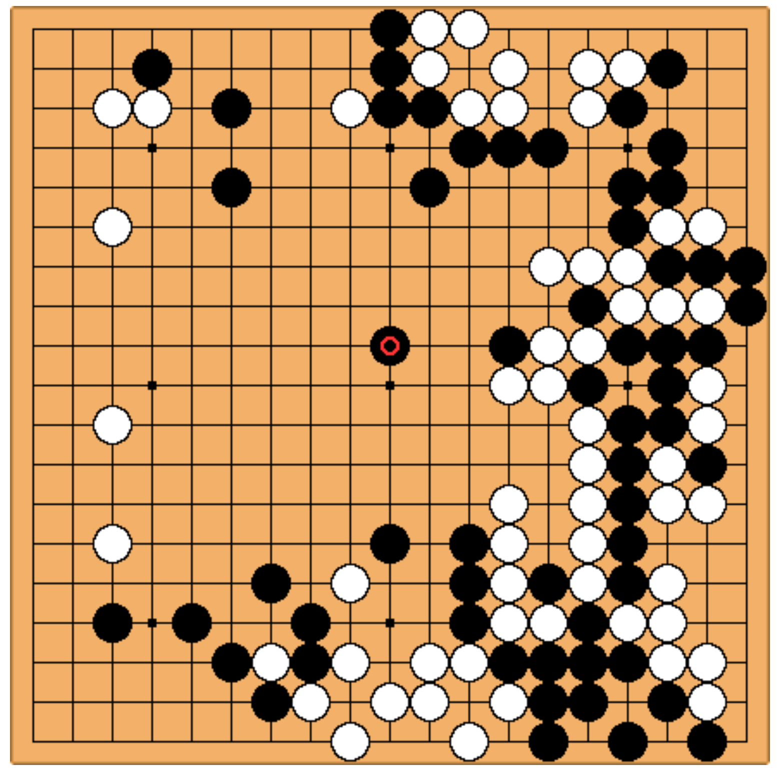


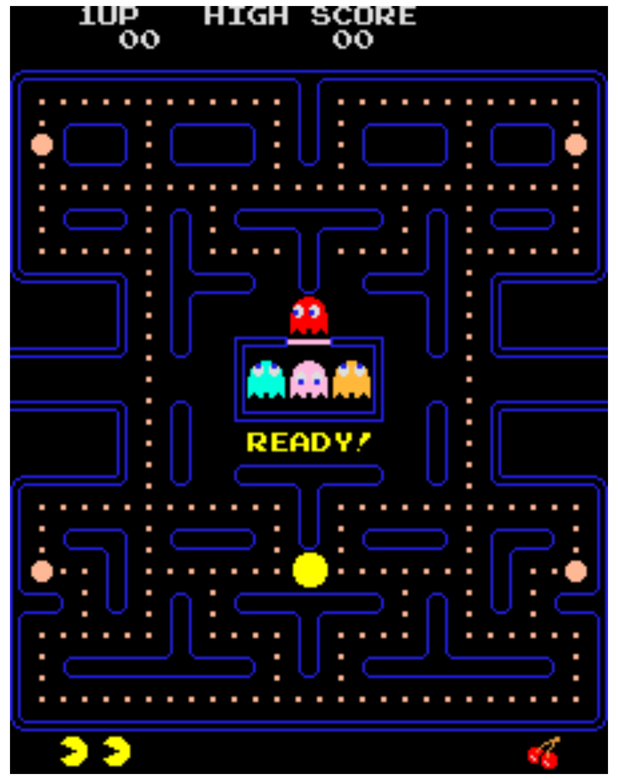


$$Q_\theta(s) = 5.2$$

The power of function approximators

- Extremely general, requires little domain knowledge

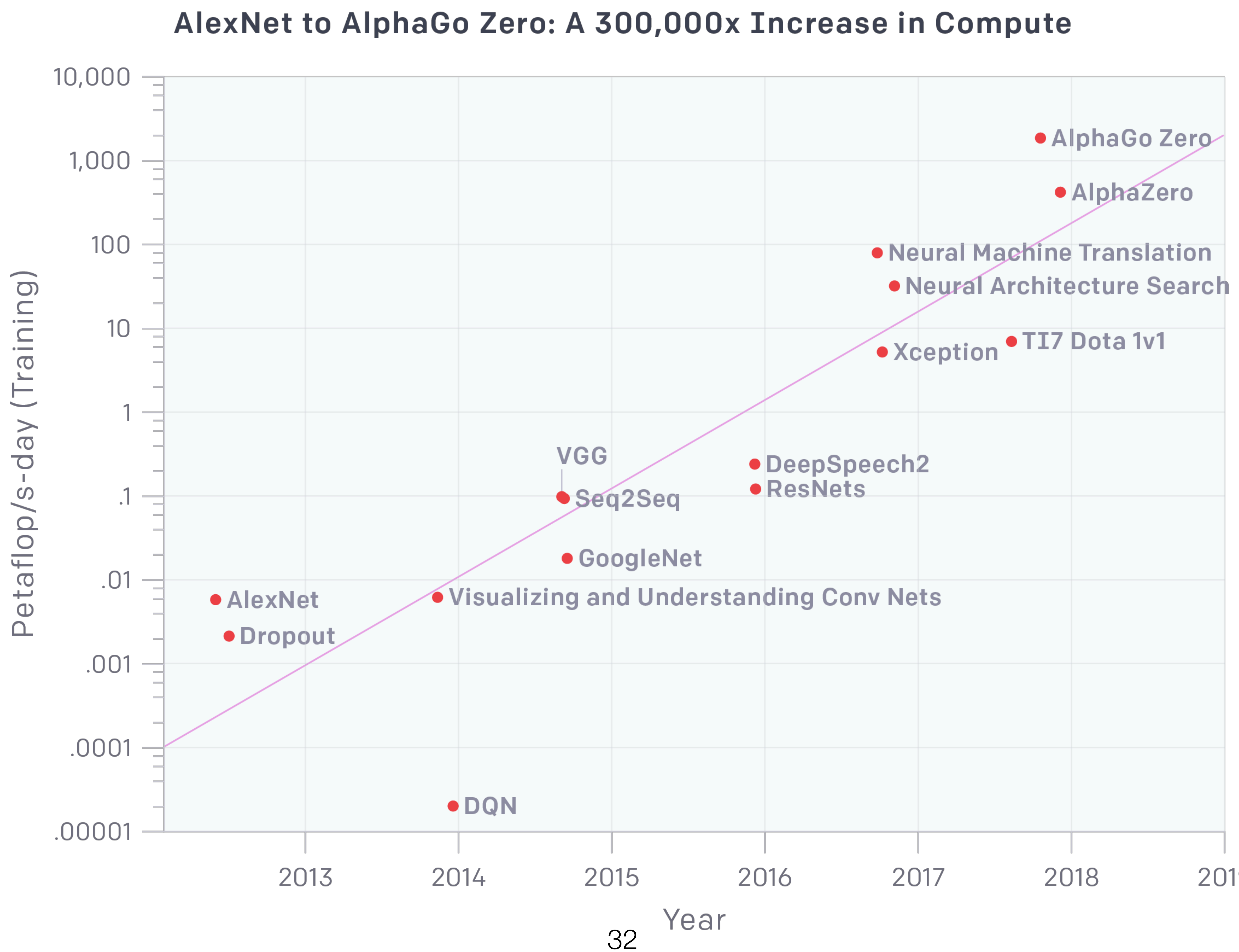




- A fundamental trade-off in machine learning:

The more complex a machine learning model,
the more data it needs to train (and not overfit).

Compute in state-of-the-art models is increasing exponentially



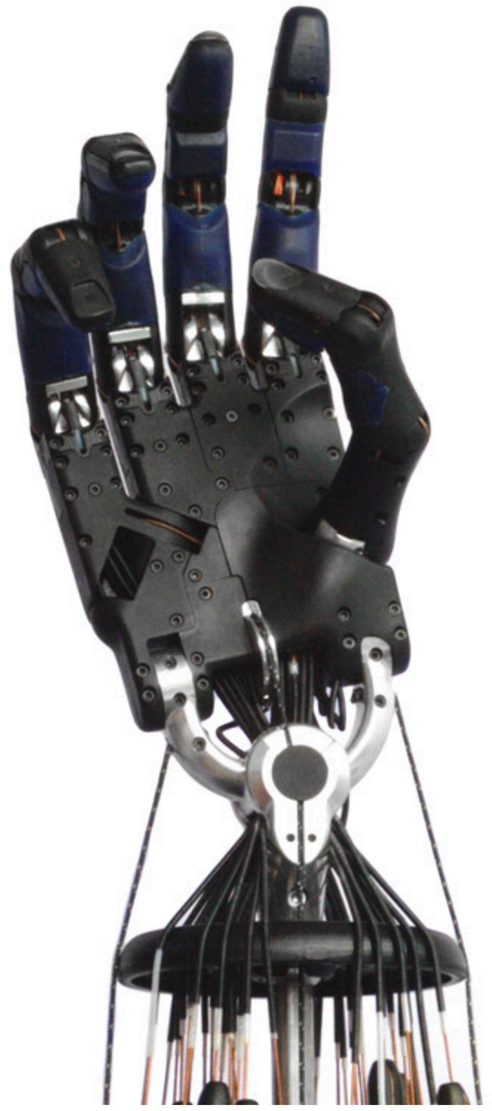
Robot experiments are expensive

- Training AlphaGo once takes over $1M in compute resources, and plays 4.9 million games. The sample complexity is enormous!

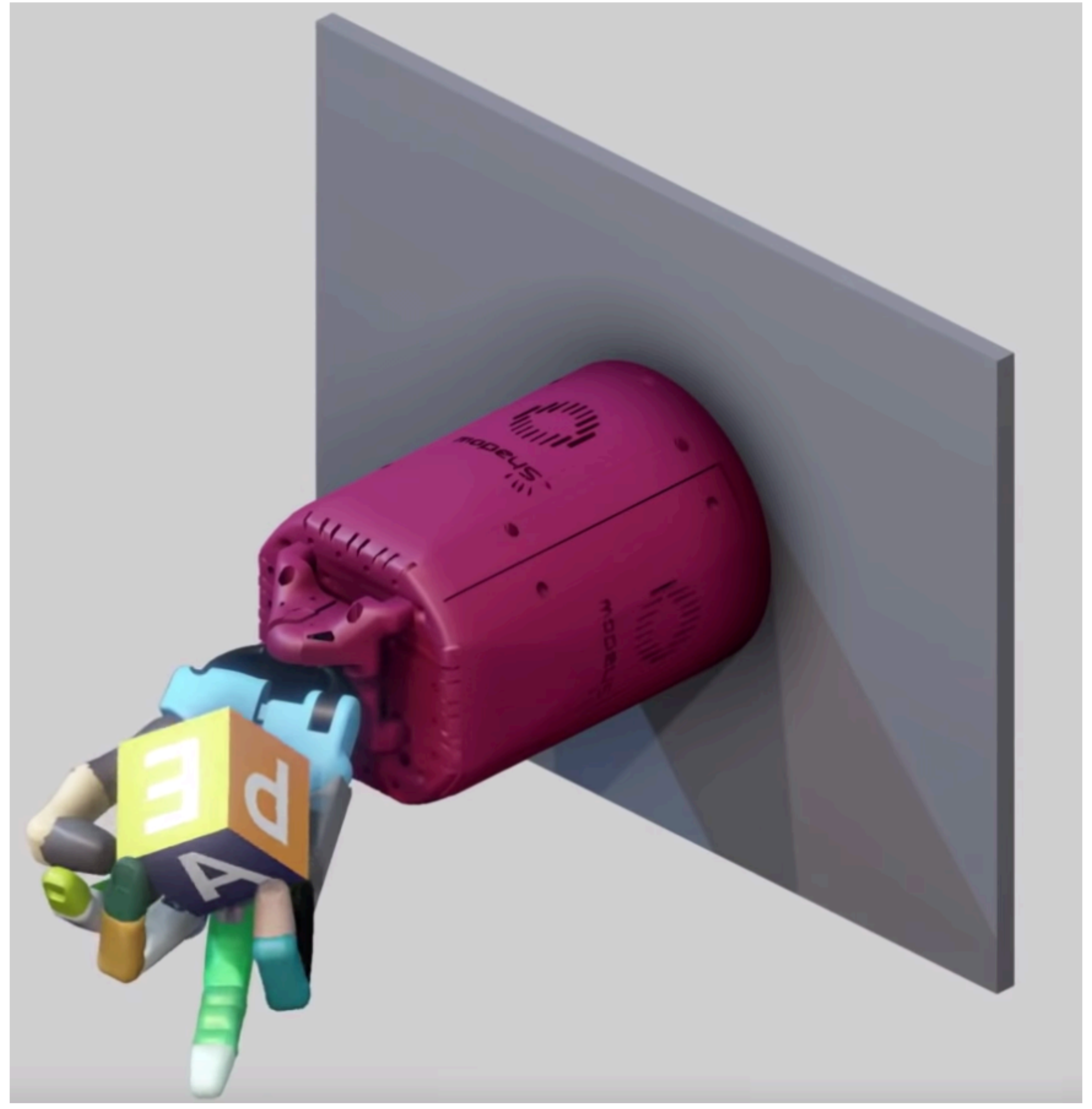
- Video games are practically free compared to the time and cost of collecting data on a robot

Robot experiments can be dangerous

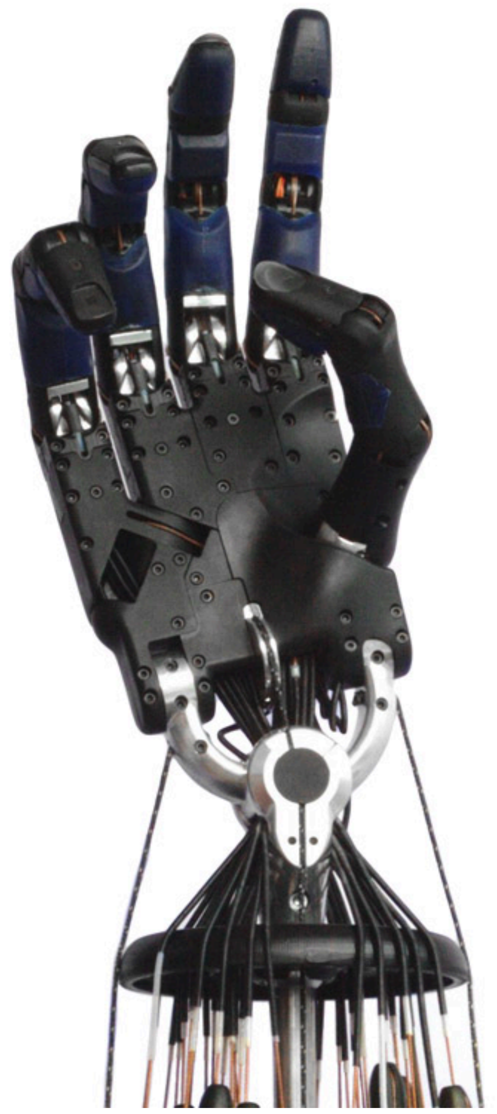
The Sim-to-Real Dilemma



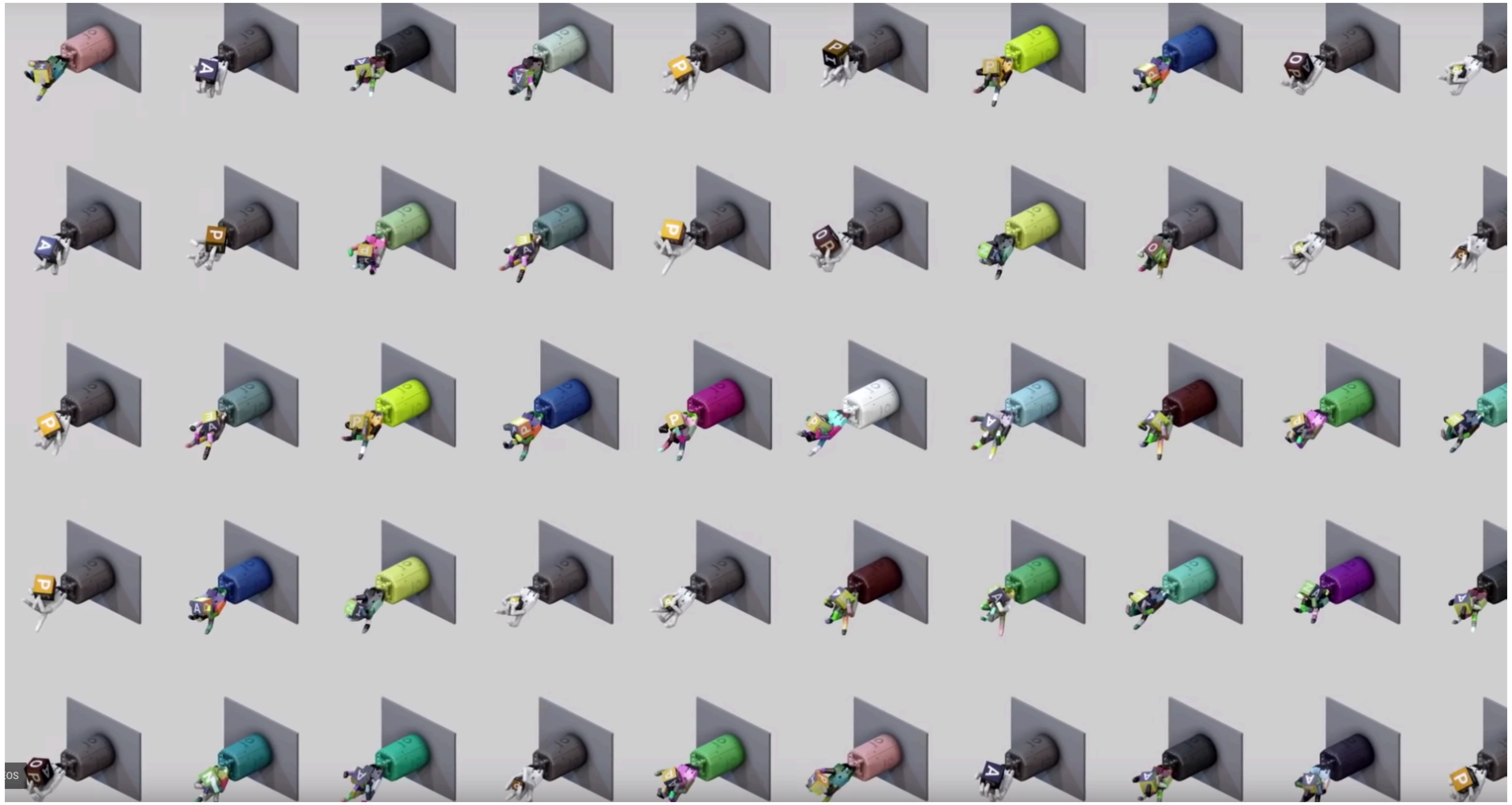
$\neq$



Robust control and the domain randomization trick



$\in$



Generality vs. Specificity

- Markov Decision Processes are an extremely general model, and Reinforcement Learning is a general purpose method for solving them.
- The more assumptions and prior knowledge you can incorporate into your model, the less you need to learn. Especially components you can accurately model.
    - Kinematics, rigid body dynamics, gravity, friction
- This often leads to specific, practical solutions (e.g. LQR)

Existing models are a powerful tool

Mujoco: https://www.youtube.com/watch?v=uRVAX_sFT24

Atari [Guo 2014]

Agent	B.Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S.Invaders
DQN	4092	168	470	20	1952	1705	581
-best	5184	225	661	21	4500	1740	1075

Deep Q-Learning

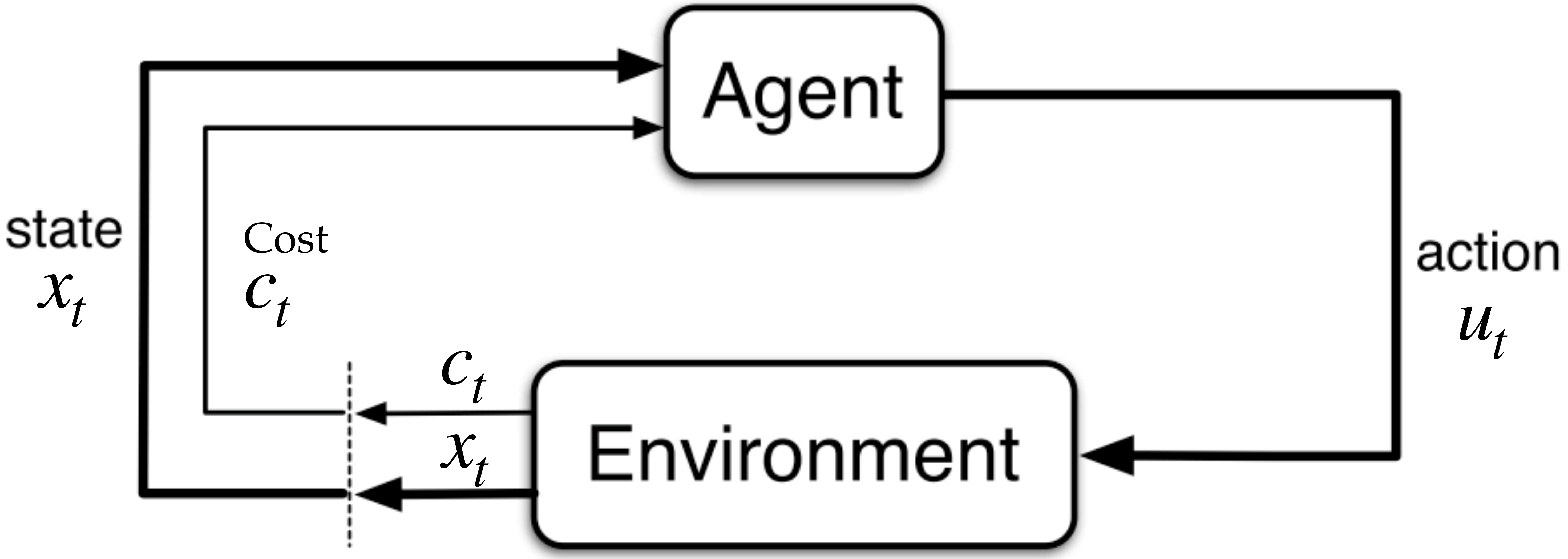
Agent	B.Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S.Invaders
UCT	7233	406	788	21	18850	3257	2354

Online Tree Search

Atlas: https://www.youtube.com/watch?v=fRj34o4hN4I

Recap

- Markov Decision Processes are a very general class of models, which encompass planning and reinforcement learning.



Recap

"Markov" means that _____ captures all information about the history $x_1, x_2, \ldots, x_t$

The most recent state $\boldsymbol{x_t}$

Recap

- The difference between planning and reinforcement learning is whether the ___________ are known

transition model / dynamics / environment

Recap

- The three general methods for reinforcement learning are...

(1) Model-based

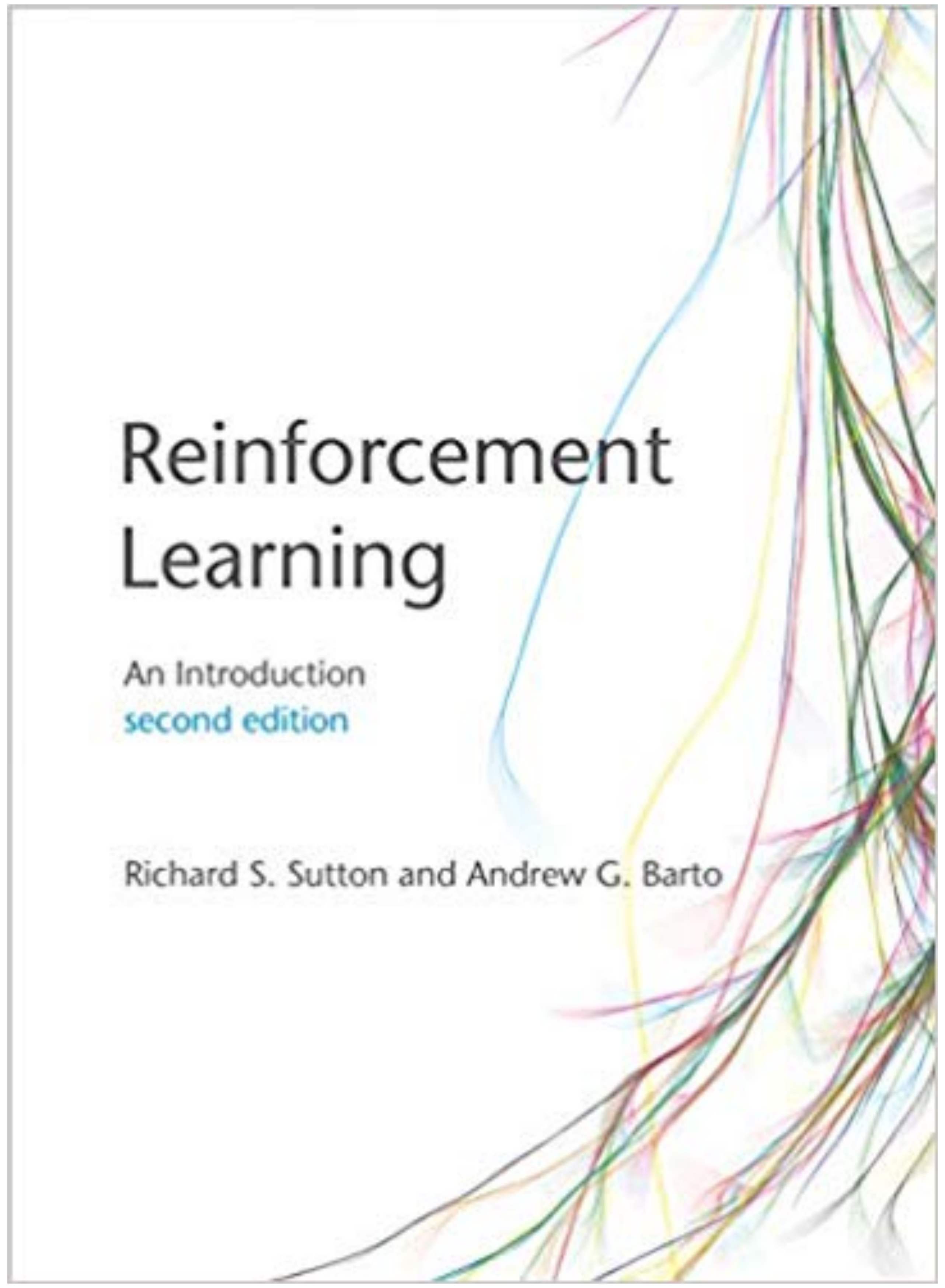
(2) Approximate dynamic programming

(3) Policy gradient

- (2) and (3) are both ________ methods

Model-free

Further Reading



The best introductory textbook on reinforcement learning you'll find anywhere. Beautifully written.
Originally written in 1998, and recently updated in 2018. Available for free online: http://incompleteideas.net/book/the-book-2nd.html

Preview of the next two lectures

Two specific instances of reinforcement learning which have already had massive practical success, or seem to show promise

- **Bandits** — what if our MDP only has a single state? (Wednesday)
- **Imitation learning** — what if we have help from a (human) expert? (Friday)