Introduction to Reinforcement Learning

CS 8803 RLR @ Gatech
Jan. 19, 2022

Jie Tan

Warm-up Exercise

How to make this cute robot run agilely?

Today's Class

1. Definition of Markov Decision Process (MDP)
2. Overview of Reinforcement Learning (RL) algorithms

Goals:

1. Learn how to formulate a RL problem
2. Understand the categories and trade-offs between different RL algorithms

Sequential Decision Problem

- Goal: select actions to maximize total future reward
- Actions may have long-term consequences
- Reward may be sparse and delayed
- It may be better to sacrifice immediate rewards for more long-term reward
- Examples:
  - Financial investment
  - Games
  - …
  - Life

Notations

- State $\mathbf{s}_t \in \mathbb{S}$: The complete information of the environment

Definition

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

  - The future is independent of the past, given the present
  - The current state capture all the information to determine the future

Notations

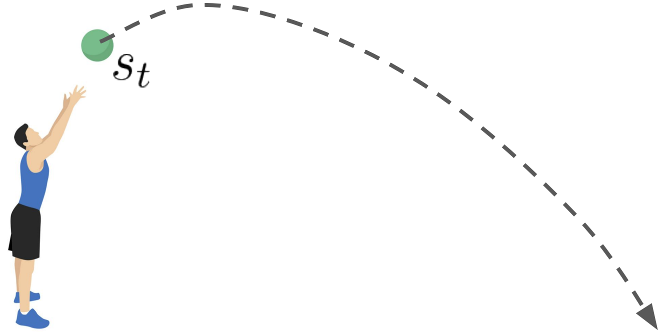
- State $\mathbf{s}_t \in \mathbb{S}$: The complete information of the environment

Definition

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

  - Quiz: What should a Markovian state be for this basketball?



A: $\mathbf{x}_t$
B: $\mathbf{x}_t$, $\mathbf{v}_t$
✓C: $\mathbf{x}_t$, $\mathbf{v}_t$, $\boldsymbol{\omega}_t$
✓D: $\mathbf{x}_t$, $\mathbf{v}_t$, $\boldsymbol{\omega}_t$, $\boldsymbol{\theta}_t$

Notations

- State $\mathbf{s}_t \in \mathbb{S}$: The complete state information of the environment
- Observation $\mathbf{o}_t \in \mathbb{O}$: The observed (potentially partial) state of the environment
  - Robotics: sensor limitations
  - Investment: not knowing everyone's portfolio on the market, unknown Monetary Policy, etc.
  - Video game: the image on the screen

Notations

- State $\mathbf{s}_t \in \mathbb{S}$: The complete state information of the environment
- Observation $\mathbf{o}_t \in \mathbb{O}$: The observed (potentially partial) state of the environment
- Action $\mathbf{a}_t \in \mathbb{A}$: The action that the agent can execute
    - Robotics: motor torque
    - Investment: buy or sell
    - Video game: joystick movement and buttons to press

Notations

- State $\mathbf{s}_t \in \mathbb{S}$: The complete state information of the environment
- Observation $\mathbf{o}_t \in \mathbb{O}$: The observed (potentially partial) state of the environment
- Action $\mathbf{a}_t \in \mathbb{A}$: The action that the agent can execute
- Reward $r_t : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{R}$: A scalar feedback signal
    - Indicates how good the current state is
    - The goal is to maximize the cumulative reward
    - Robotics (humanoid robot walking):
        - + reward for moving forward
        - - reward for losing balance
    - Investment
        - + reward for $ in the bank
    - Video game
        - +/- reward based on the score change on the screen

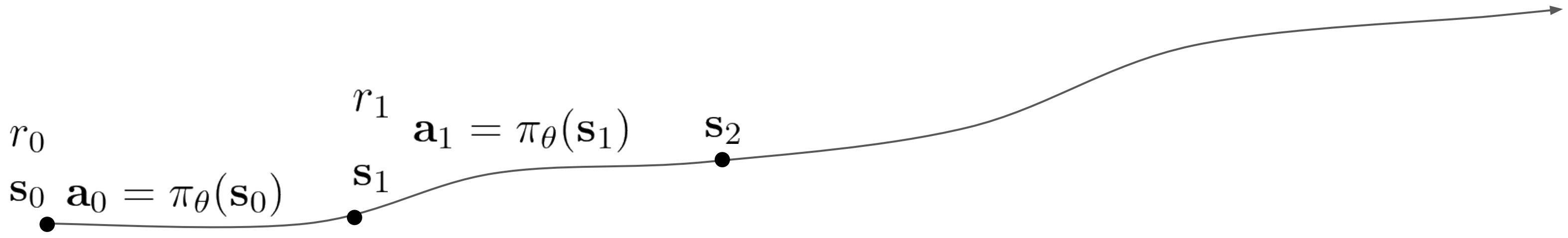
Notations

- State $\mathbf{s}_t \in \mathbb{S}$: The complete state information of the environment
- Observation $\mathbf{o}_t \in \mathbb{O}$: The observed (potentially partial) state of the environment
- Action $\mathbf{a}_t \in \mathbb{A}$: The action that the agent can execute
- Reward $r_t : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{R}$: A scalar feedback signal
- Transition function $\mathcal{T} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \mapsto \mathbb{R}$: A rule that governs how the state evolve in the environment
    - $\mathcal{T}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) = \mathbb{P}[\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t]$
    - Robotics: Physics
    - Investment: Behavioral economics, macroeconomics, etc.
    - Video game: Game rule

Notations

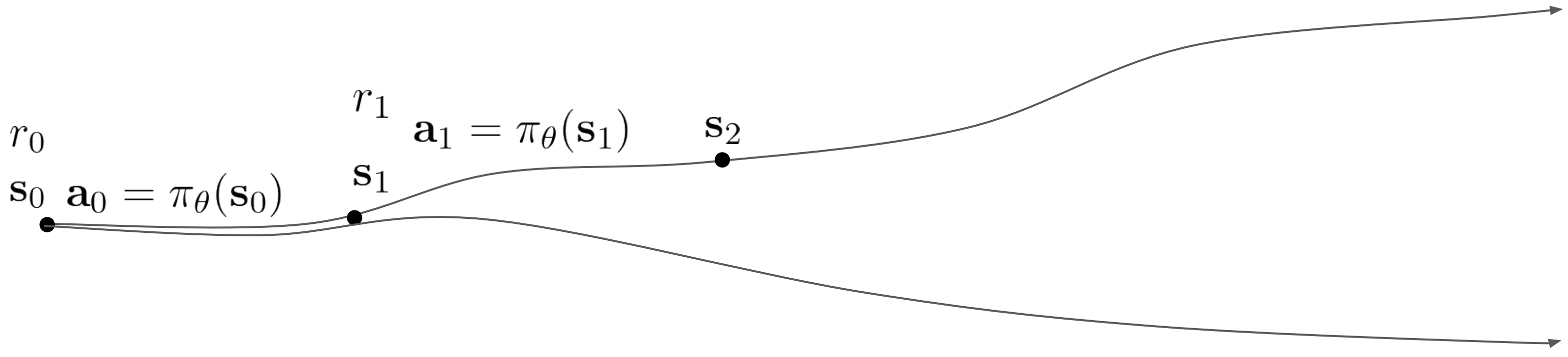
- State $\mathbf{s}_t \in \mathbb{S}$: The complete state information of the environment
- Observation $\mathbf{o}_t \in \mathbb{O}$: The observed (potentially partial) state of the environment
- Action $\mathbf{a}_t \in \mathbb{A}$: The action that the agent can execute
- Reward $r_t : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{R}$: A scalar feedback signal
- Transition function $\mathcal{T} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \mapsto \mathbb{R}$: A rule that governs how the state evolve in the environment
- Policy $\pi_\theta : \mathbb{S} \mapsto \mathbb{A}$: The feedback decision function
    - Chooses proper actions based on the current situation (state/observation)
    - θ is the parameter of the policy: e.g. weight of a neural network

Markov Decision Process



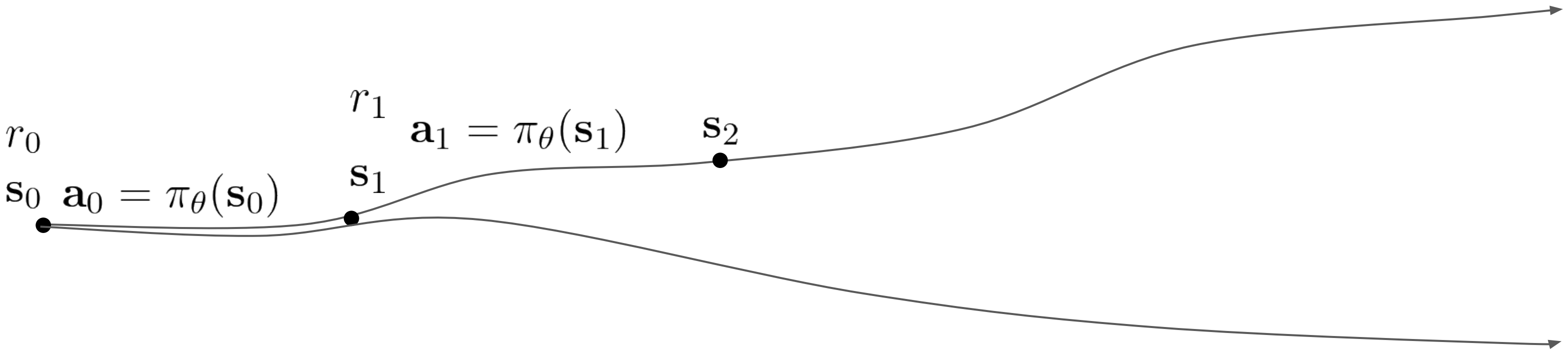
$$J(\theta) = \sum_t r_t$$

Markov Decision Process



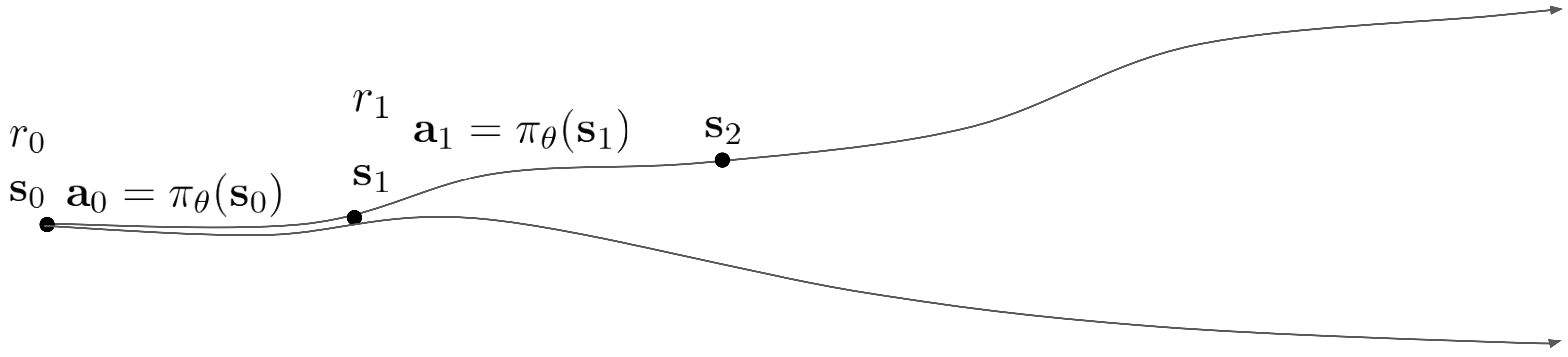
$$J(\theta) = \sum_t r_t$$

Markov Decision Process



$$J(\theta) = \mathbb{E}_\theta[\sum_t r_t]$$

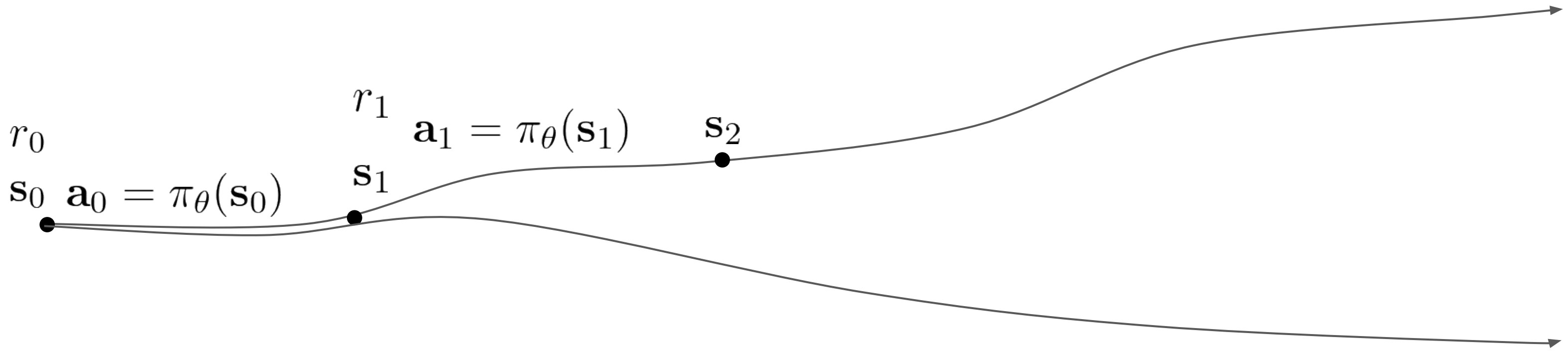
Markov Decision Process



$$J(\theta) = \mathbb{E}_\theta[\sum_t \gamma^t r_t]$$

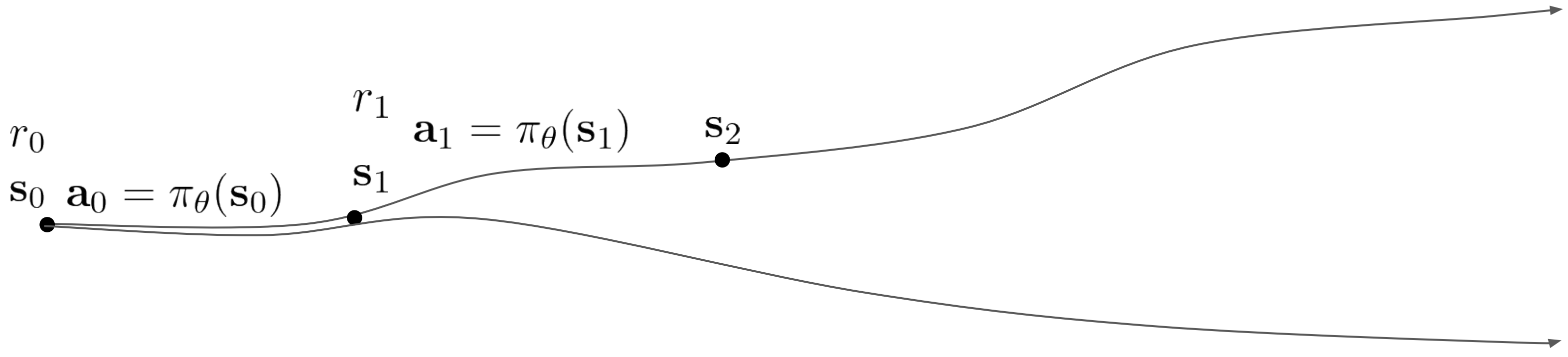
discount factor

Markov Decision Process



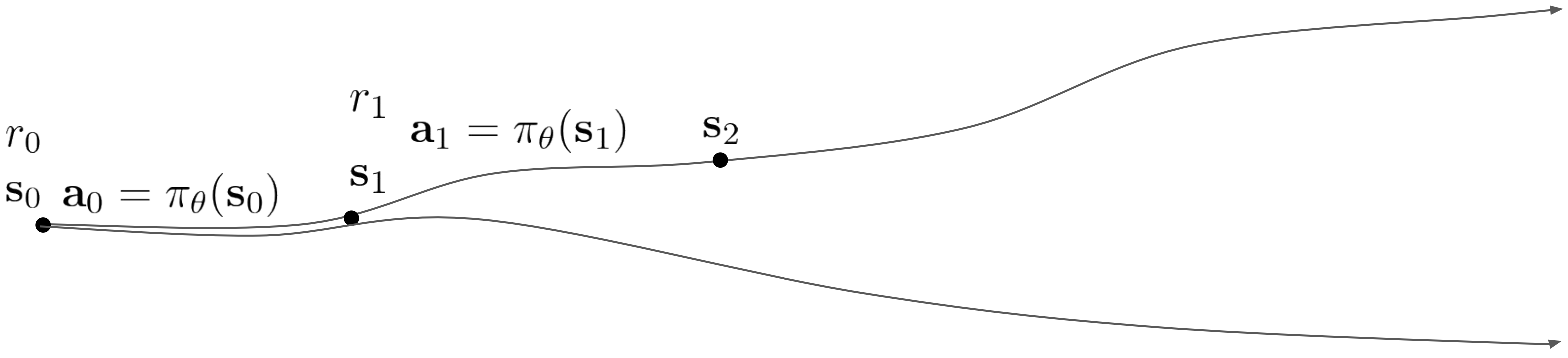
$$J(\theta) = \mathbb{E}_\theta[\underbrace{\sum_t \gamma^t r_t}_{\text{return}}]$$

Markov Decision Process



$$J(\theta) = \underbrace{\mathbb{E}_\theta[\sum_t \gamma^t r_t]}_{\text{expected return}}$$

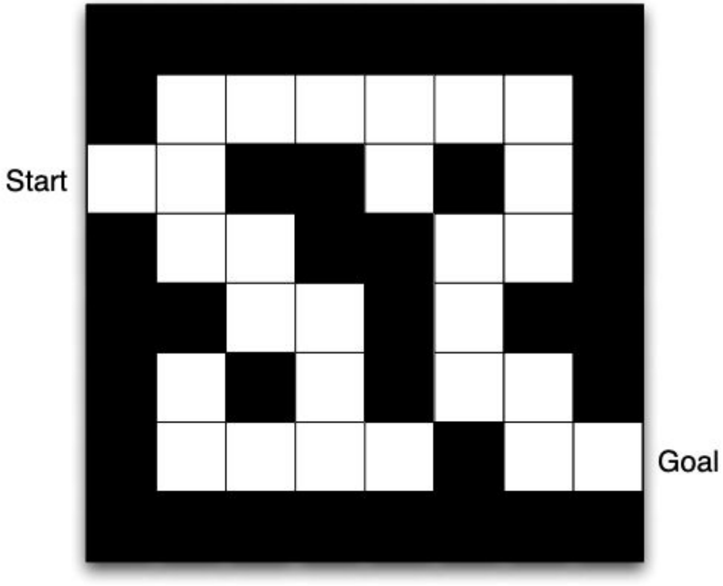
Markov Decision Process



$$\pi_\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \mathbb{E}[\sum_t \gamma^t r_t]$$

Example 1. Maze

Goal: Traverse the maze as fast as possible



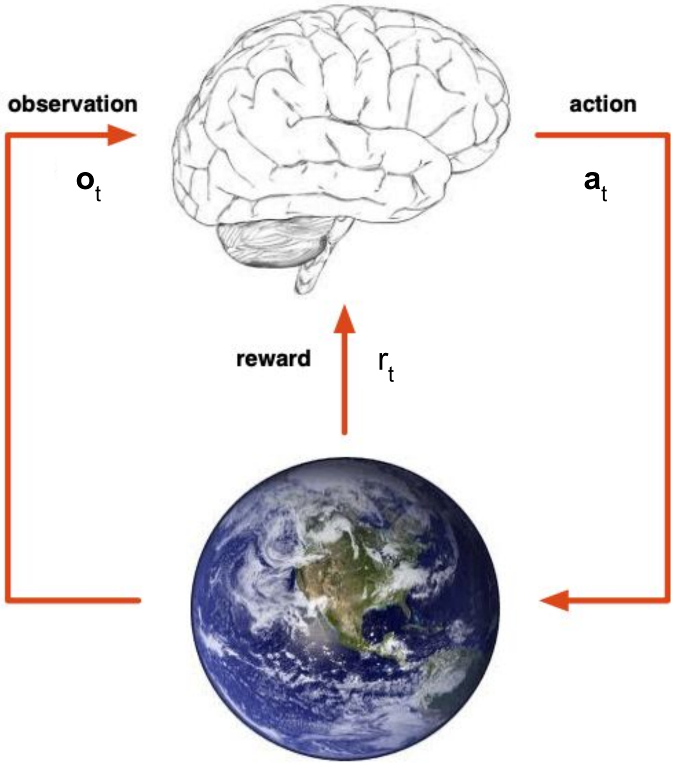
- State: agent's location
- Action: N,W,E,S
- Reward: -1 for each step
- Transition: move to the adjacent location
- Policy: lookup table

Example 2. Robot Locomotion

Goal: Run as fast as possible

- Observation: joint angles, roll, pitch, yaw
- Action: motor torques
- Reward: $r = (\mathbf{p}_n - \mathbf{p}_{n-1}) \cdot \mathbf{d}$
- Transition: equation of motion
- Policy: neural network

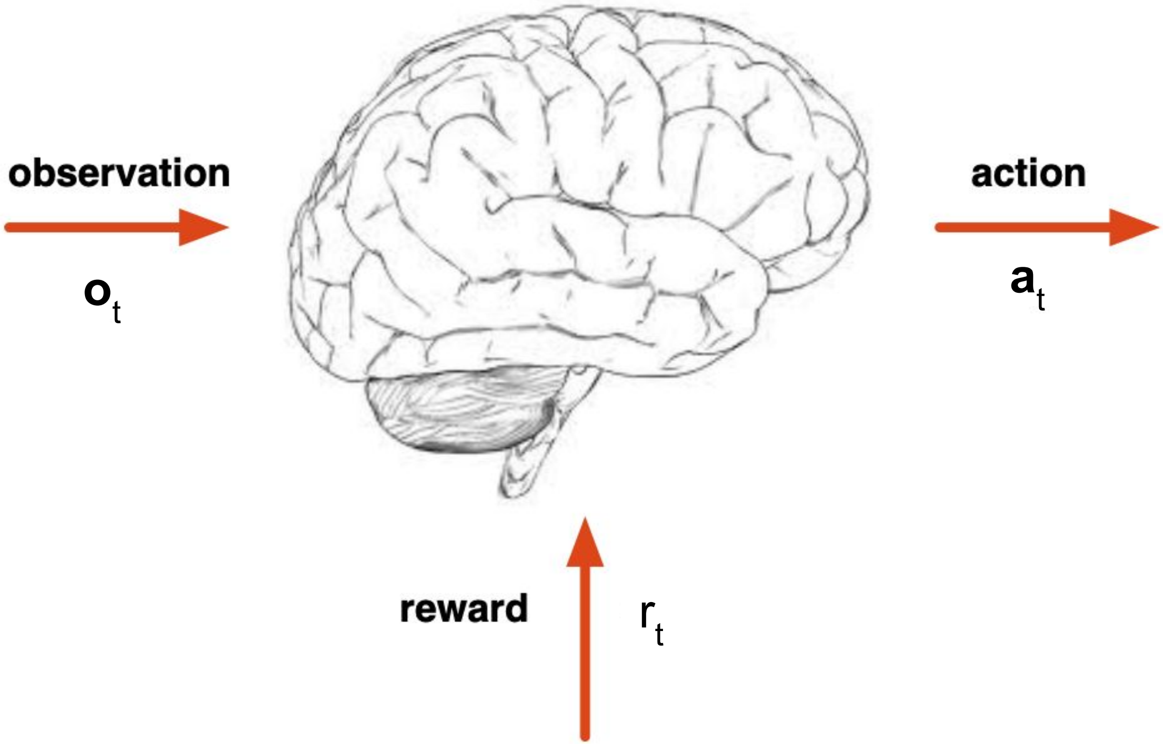
Reinforcement Learning: Agent and Environment



For each time step t

- Agent
  - receives observation $\mathbf{o}_t$
  - executes action $\mathbf{a}_t$
  - receives reward r_t
- Environment
  - receives action $\mathbf{a}_t$
  - emits observation $\mathbf{o}_{t+1}$
  - emits reward r_{t+1}

Inside an Agent



Inside an Agent

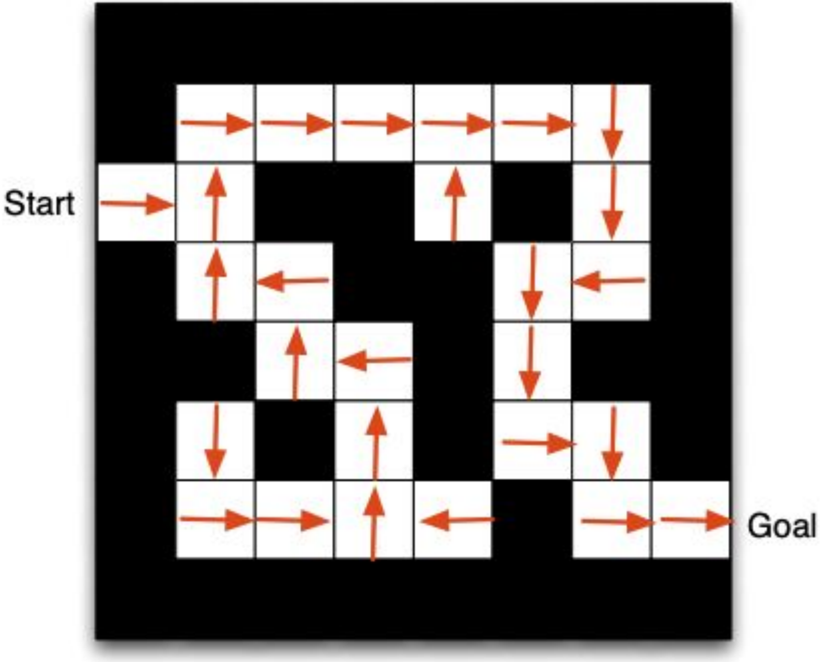
An agent consists one or more of the following components:

- Policy: What to do in the current situation?
- Value function: How good is each state / action?
- Model: What’s going to happen if this action is taken?

Policy

Mapping from state to action

- Deterministic: $a = \pi(s)$
- Stochastic: $\pi(a|s) = \mathbb{P}(a_t = a|s_t = s)$
- Representation
  - Look-up table
  - Neural network
  - …

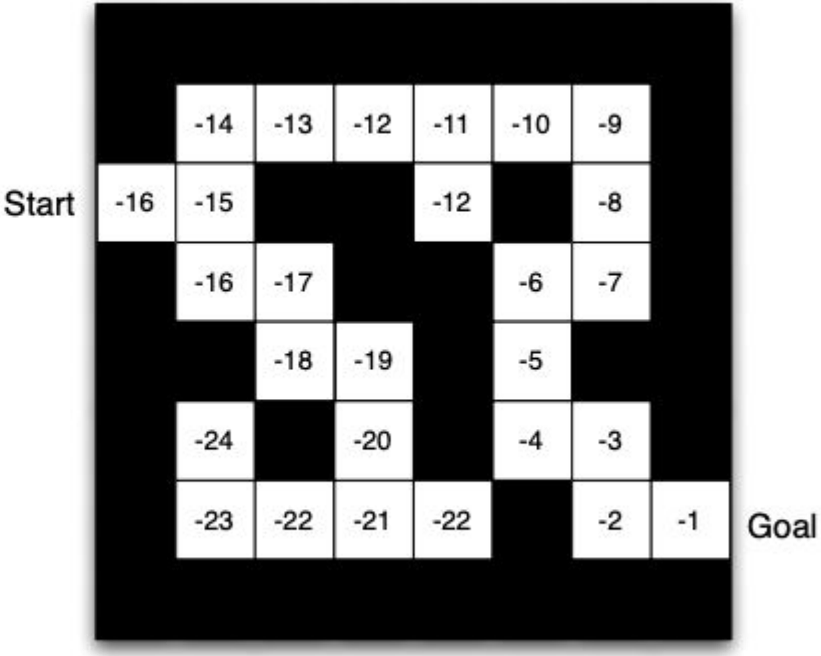


Value Function

Prediction of accumulated future reward

$$V_\pi(s) = \mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s]$$

- Used to evaluate states
- And select actions

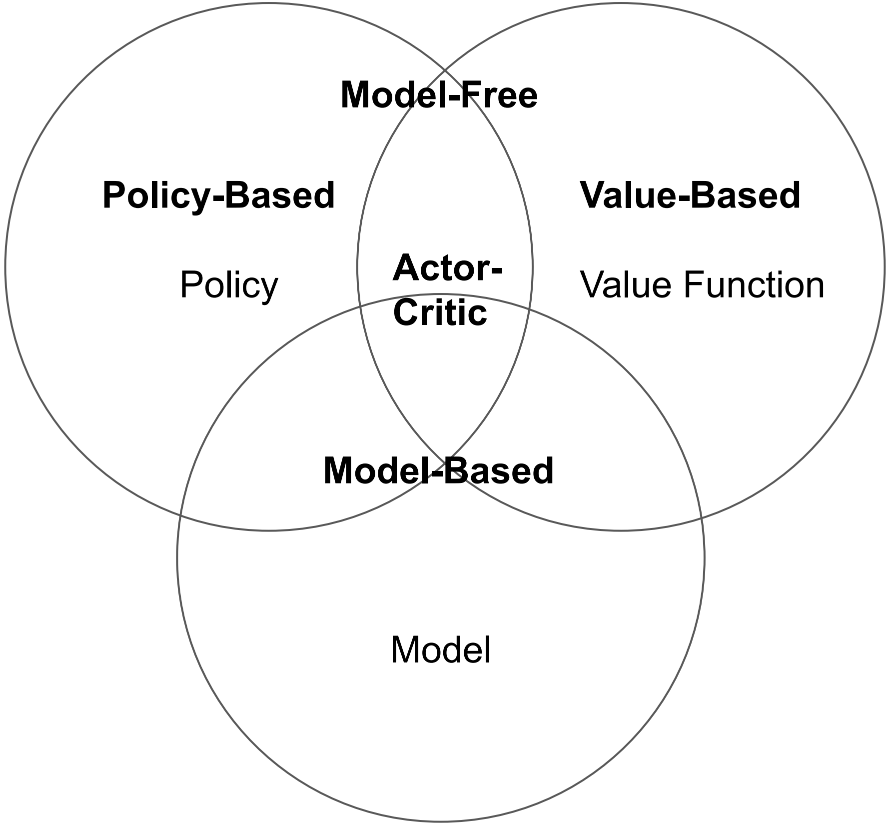


Model

Prediction of the environment

- Predicts the next state
  $$\mathbb{P}[s_{t+1} = s' | s_t = s, a_t = a]$$
- Predicts the reward
  $$\mathbb{E}[r_{t+1} | s_t = s, a_t = a]$$
- Used to “imagine” the potential consequences to choose the optimal action

RL Algorithms Taxonomy



Example of RL Algorithms

- Policy-based
  - REINFORCE
  - Natural policy gradient
  - Proximal Policy Optimization (PPO)
- Value-based
  - Q-learning, Deep Q network (DQN)
  - QT-OPT
- Actor-critic
  - Asynchronous advantage actor-critic (A3C)
  - Soft actor-critic (SAC)
- Model-based
  - Dyna
  - Probabilistic ensembles with trajectory sampling (PETS)

RL Algorithms in Nutshell

Current iteration

Step 1

Exploration

Execute the policy and add randomness to the policy

Step 2

Evaluation

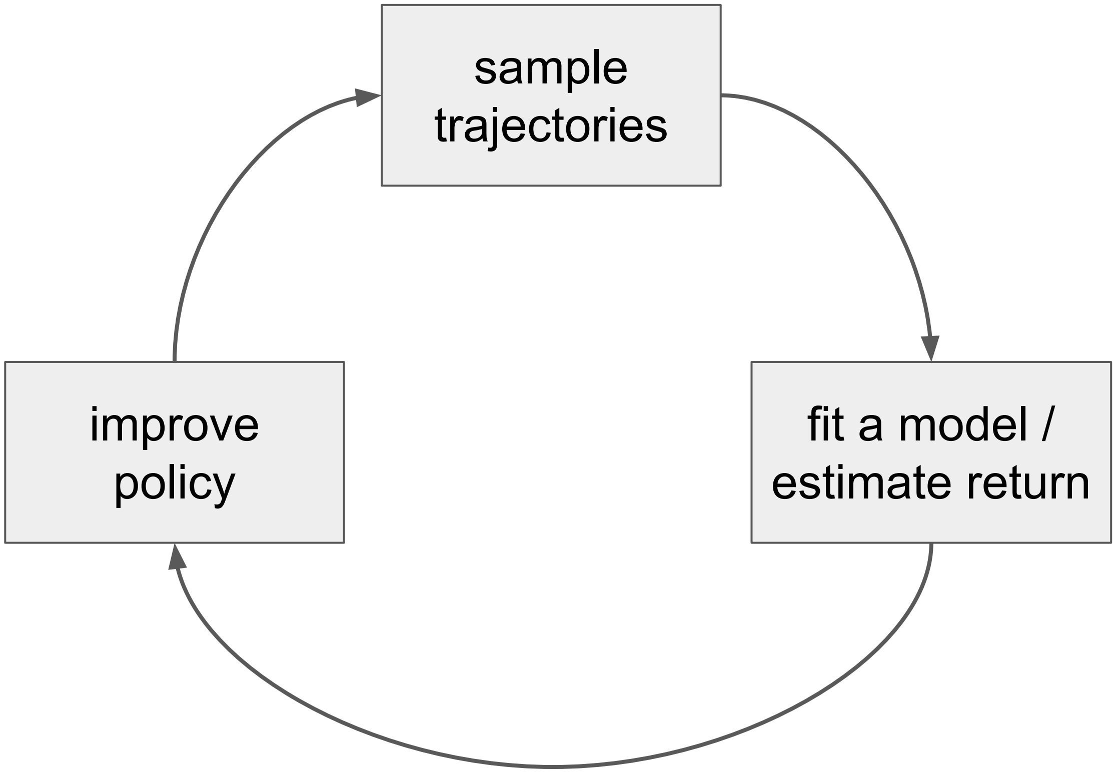
How does the policy perform?

Step 3

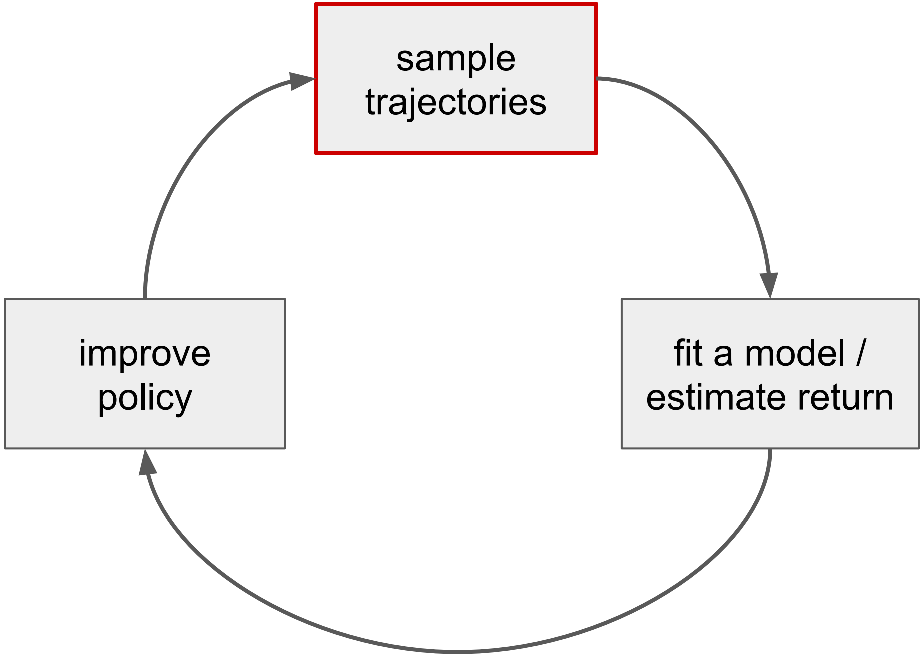
Exploitation

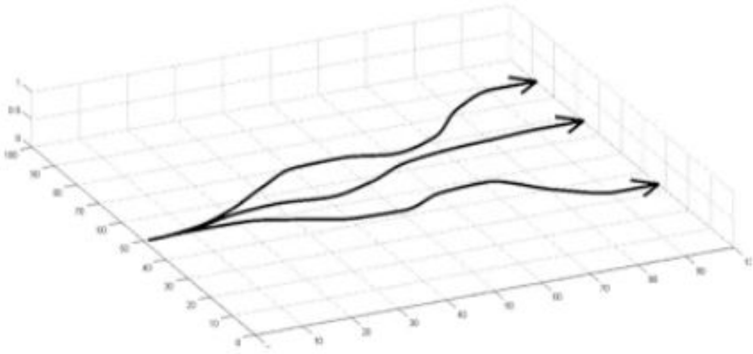
If the result is better than expected, do the same more often in the future

Three Components of RL Algorithms

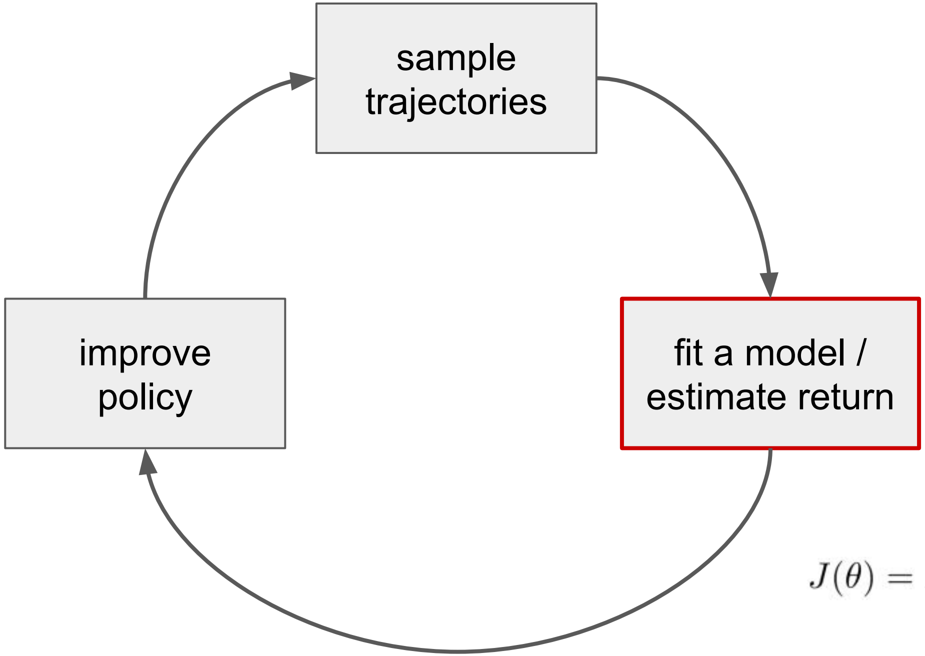


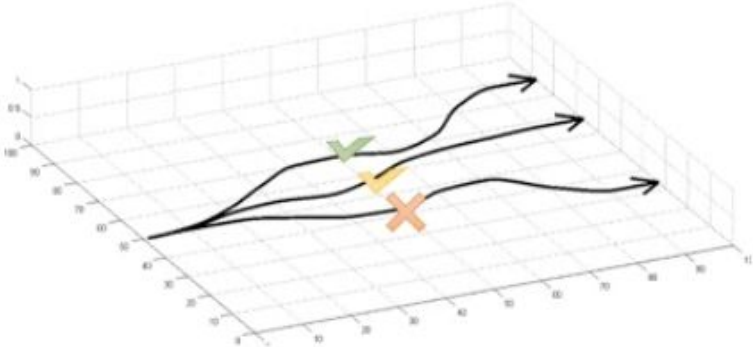
Example 1: Policy Gradient





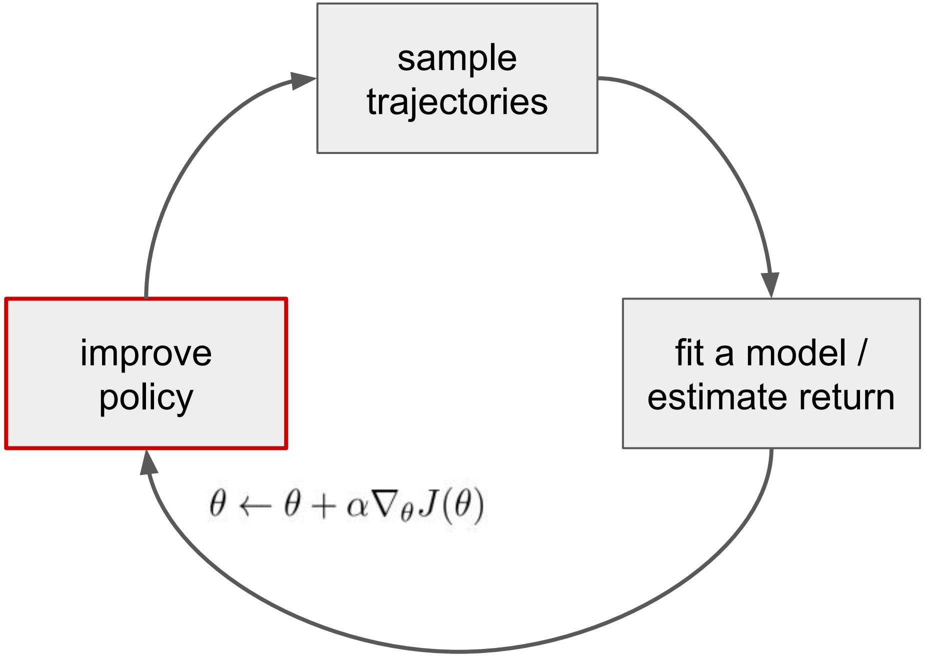
Example 1: Policy Gradient

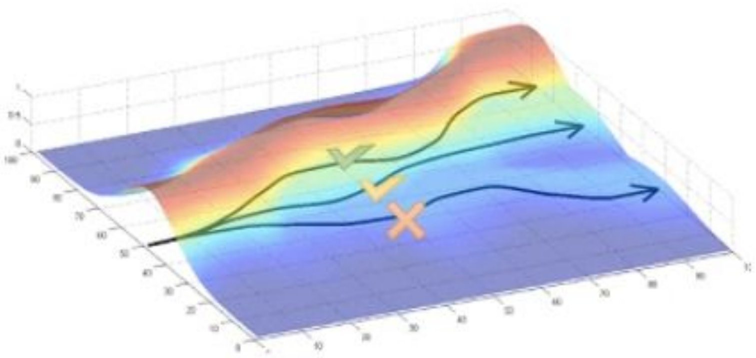




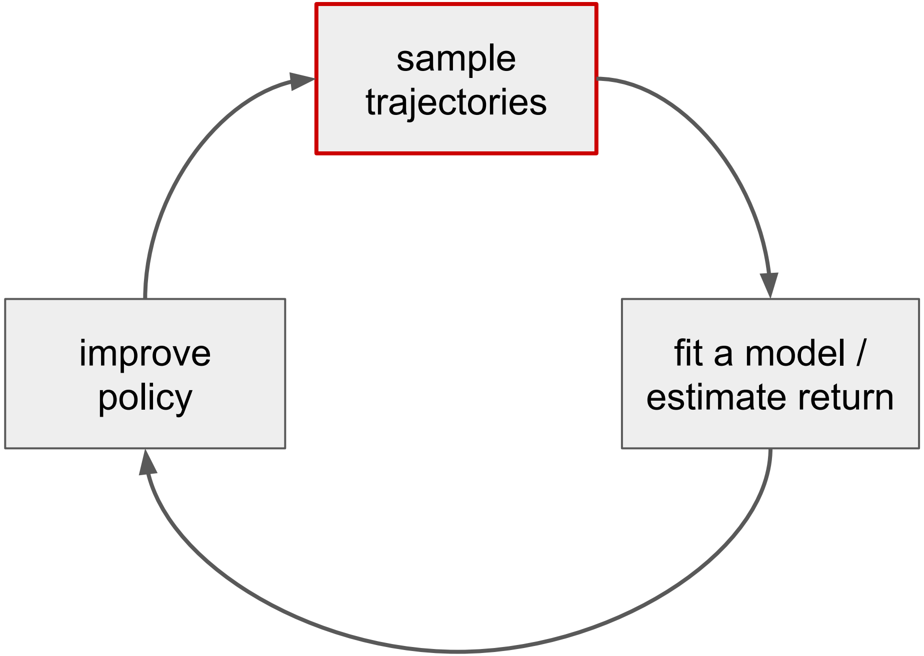
$$J(\theta) = E_\pi\left[\sum_t r_t\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_t r_t^i$$

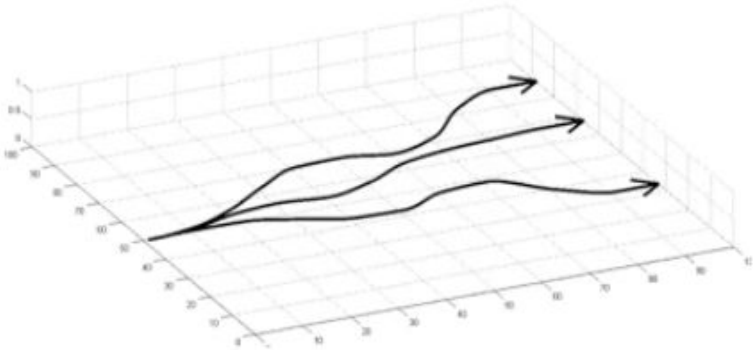
Example 1: Policy Gradient



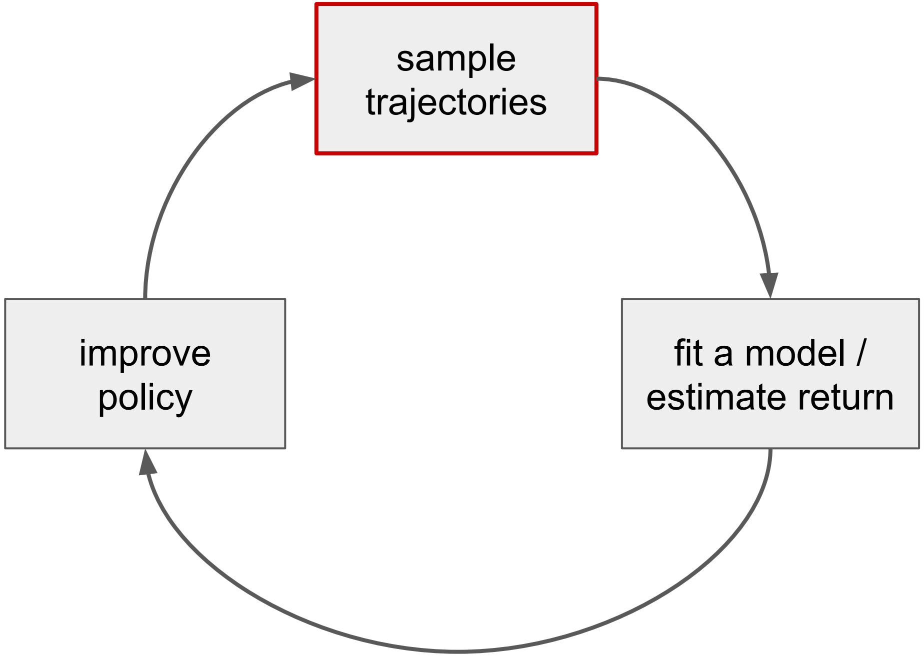


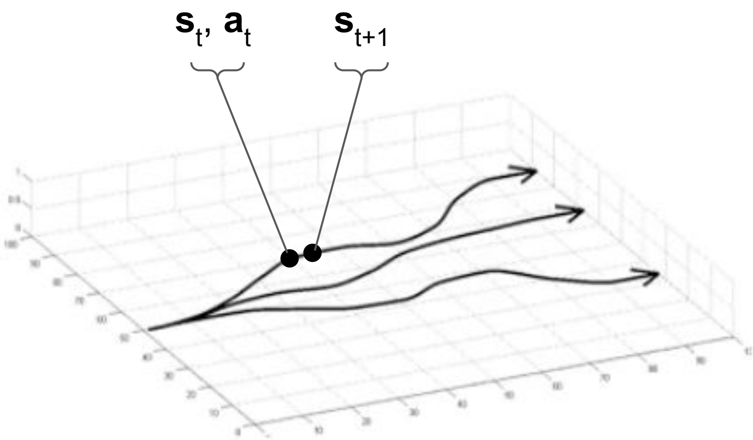
Example 2: Model-Based



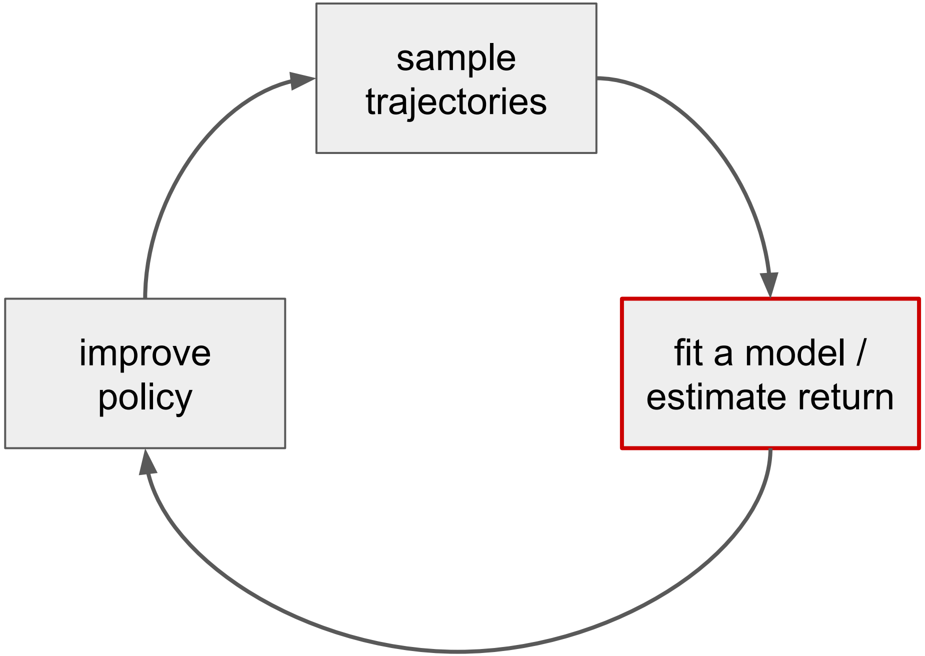


Example 2: Model-Based

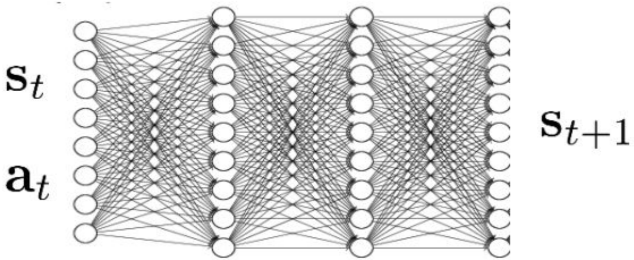




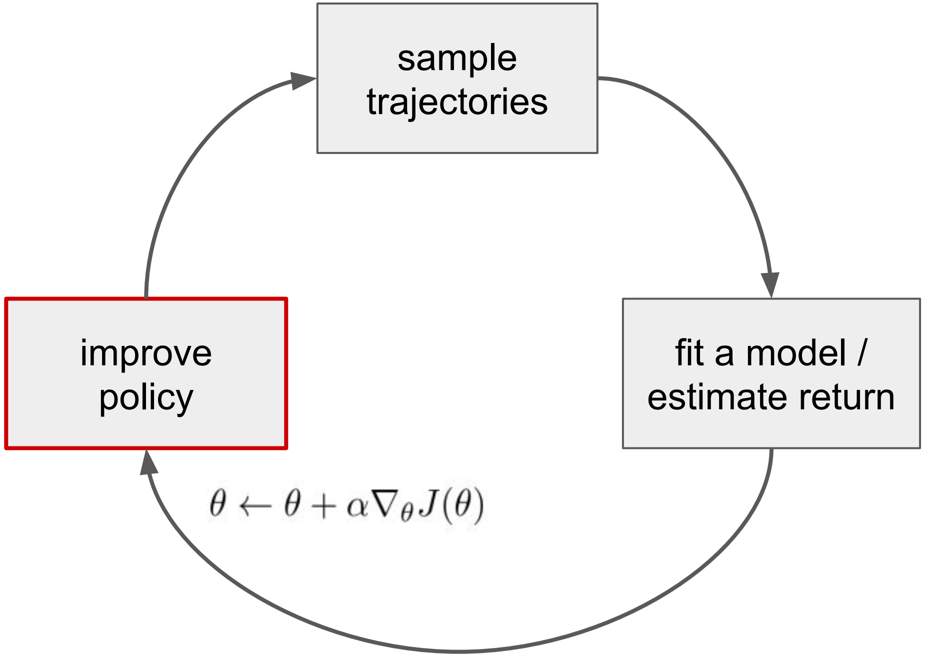
Example 2: Model-Based



$$\text{learn } f_\phi \text{ such that } \mathbf{s}_{t+1} \approx f_\phi(\mathbf{s}_t, \mathbf{a}_t)$$



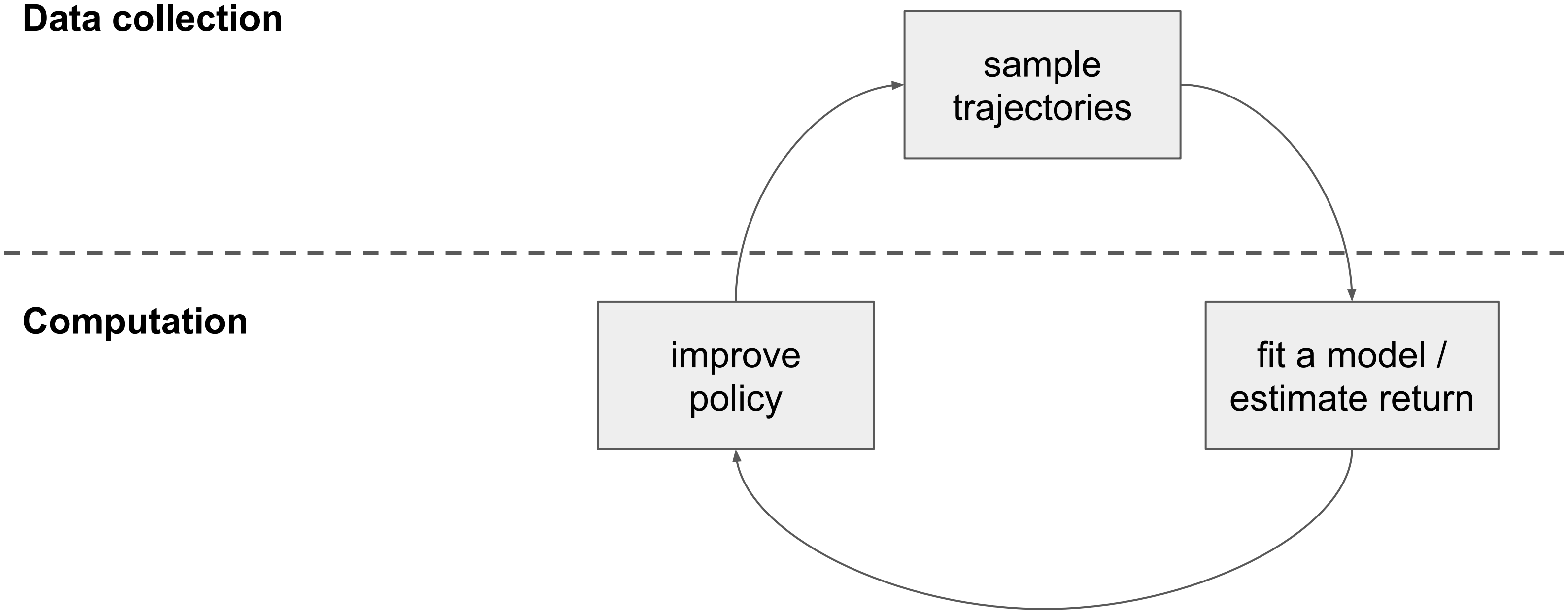
Example 2: Model-Based



backprop through f_ϕ and r to train $\pi_\theta(\mathbf{s}_t) = \mathbf{a}_t$

Trade-off between Algorithms

Which component is more expensive?



Trade-off between Algorithms

Which component is more expensive?

Data collection

- Real world (robot / power grid): expensive
- Simulator: fast

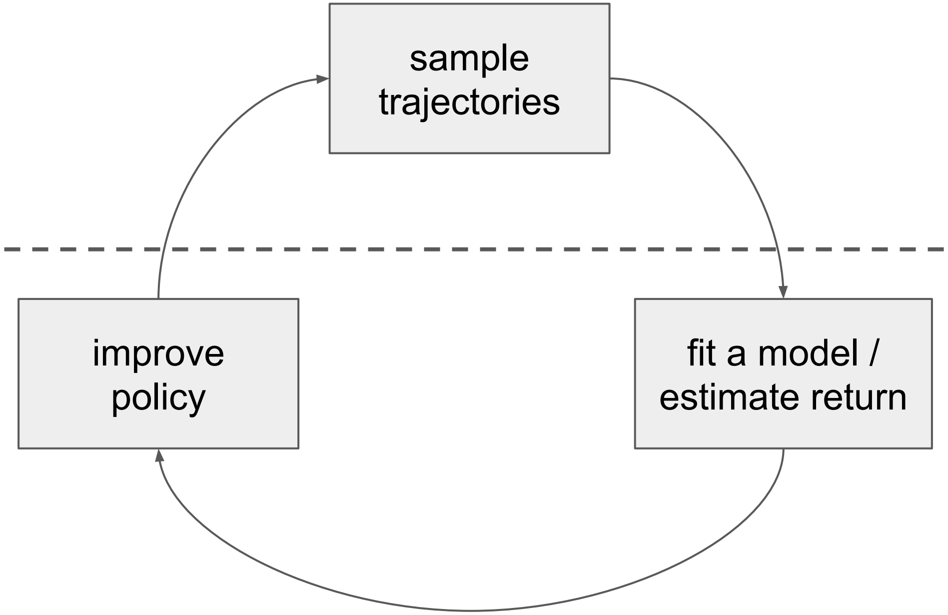
Computation

- Policy gradient: fast

$$J(\theta) = E_\pi\left[\sum_t r_t\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_t r_t^i$$

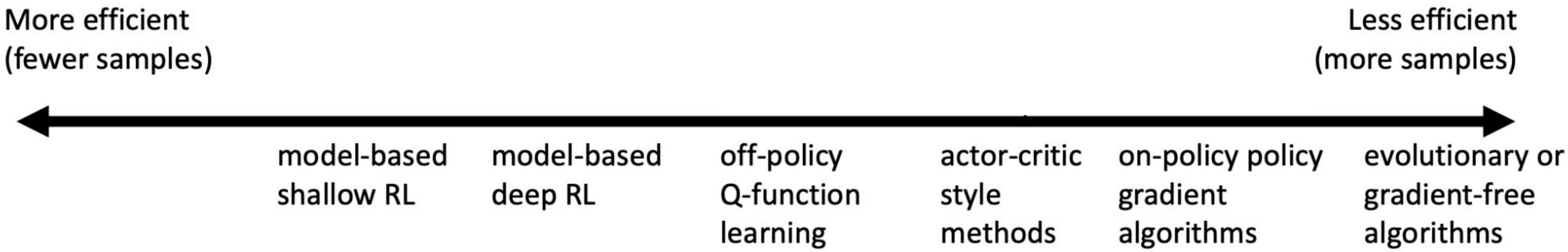
- Model-based: expensive

$$\text{learn } \mathbf{s}_{t+1} \approx f_\phi(\mathbf{s}_t, \mathbf{a}_t)$$



Trade-off between Algorithms

- Sample efficiency



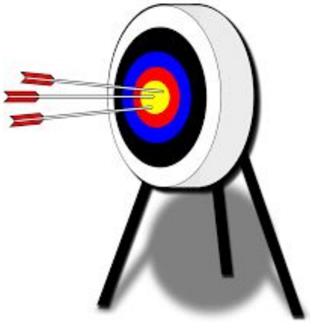
Quiz: Why do we care about ***less*** efficient algorithms?

Answer: Less efficient algorithms are often more parallel and take less training time!

Trade-off between Algorithms

- Sample efficiency
- Stability and ease-of-use
    - Converge? Hopefully.
    - Converge to what? Not sure.
    - Converge everytime? Probably not.

Fixed target: dataset



Supervised learning

Moving target: policy rollouts

Policy

Reinforcement learning

Additional Resources

- (Book) Reinforcement Learning: An Introduction, 2nd Edition, Sutton R. & Barto A. (pdf), Chapter 1 & 3
- (Online class) Introduction to Reinforcement Learning with David Silver (website), Lecture 1 & 2
- (Online class) Deep Reinforcement Learning @ UC Berkeley, Lecture 4 (video)