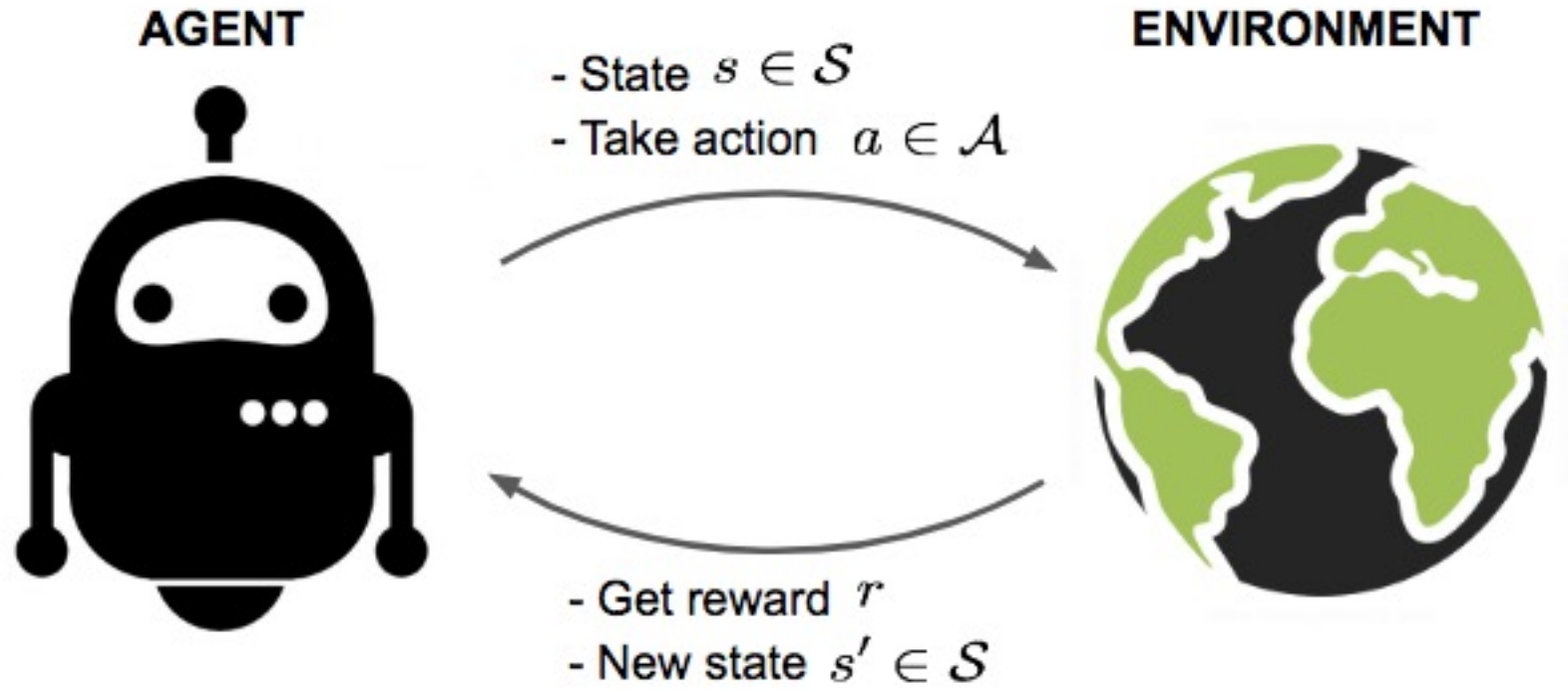
| Week | Date | Topic | Note |
|---|---|---|---|
| 1 | **2/14~2/18** | Introduction | No class on 2/18 |
| 2 | **2/21~2/25** | Machine Learning I | |
| 3 | **2/28~3/4** | Machine Learning II | HW1 announce (3/1) |
| 4 | **3/7~3/11** | Problem Solving by Searching | |
| 5 | **3/14~3/18** | Adversarial Search | HW1 due (3/15) and HW2 announce (3/18) |
| 6 | **3/21~3/25** | Markov Decision Process | |
| 7 | **3/28~4/1** | Reinforcement Learning | HW2 due and HW3 announce (4/1) |
| 8 | **4/4~4/8** | Spring Break | |
| 9 | **4/11~4/15** | Constraint Satisfaction Problems | HW3 due and Final project announce (4/12) |
| 10 | **4/18~4/22** | Bayesian Network | HW4 announcement (4/19) |
| 11 | **4/25~4/29** | Knowledge, Reasoning, and Planning | Final project proposal due (4/26) |
| 12 | **5/2~5/6** | 3D Computer Vision | HW4 due and HW5 announce (5/3) |
| 13 | **5/9~5/13** | Robot Navigation | |
| 14 | **5/16~5/20** | Intelligent Driving Systems | HW5 due (5/17) |
| 15 | **5/23~5/27** | Guest Talk (TBA) | Final project checkpoint report (5/24)<br>No class on 5/27 |
| 16 | **5/30~6/3** | Guest Talk (TBA) | No class on 6/3 |
| 17 | **6/6~6/10** | No class | |
| 18 | **6/13~6/17** | ***Final Project Demo*** | Final video and report due (6/14) |




# Reinforcement Learning

## Spring 2022

Yi-Ting Chen

https://lilianweng.github.io/posts/2018-02-19-rl-overview/

# AlphaGo, AlphaZero

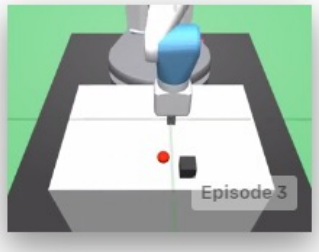
- Deep Mind, 2016+

# OpenAI Gym

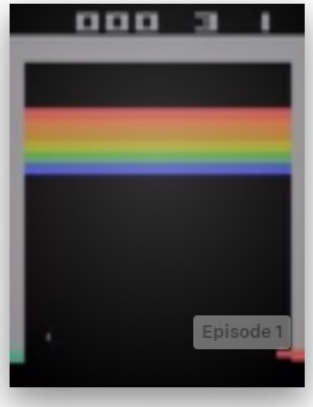
- Benchmark problems for learning agents


Acrobot-v1
Swing up a two-link robot.


Ant-v2
Make a 3D four-legged robot walk.



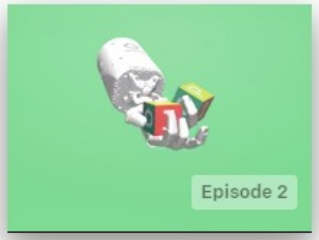

FetchPush-v0
Push a block to a goal position.



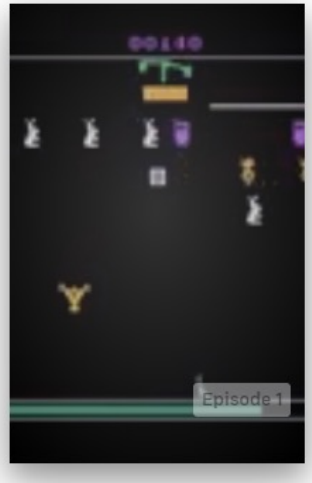

Breakout-ram-v0
Maximize score in the game Breakout, with RAM as input


MountainCarContinuous-v0
Drive up a big hill with continuous control.


Humanoid-v2
Make a 3D two-legged robot walk.




HandManipulateBlock-v0
Orient a block using a robot hand.




Carnival-v0
Maximize score in the game Carnival, with screen images as input

https://gym.openaicom/envs

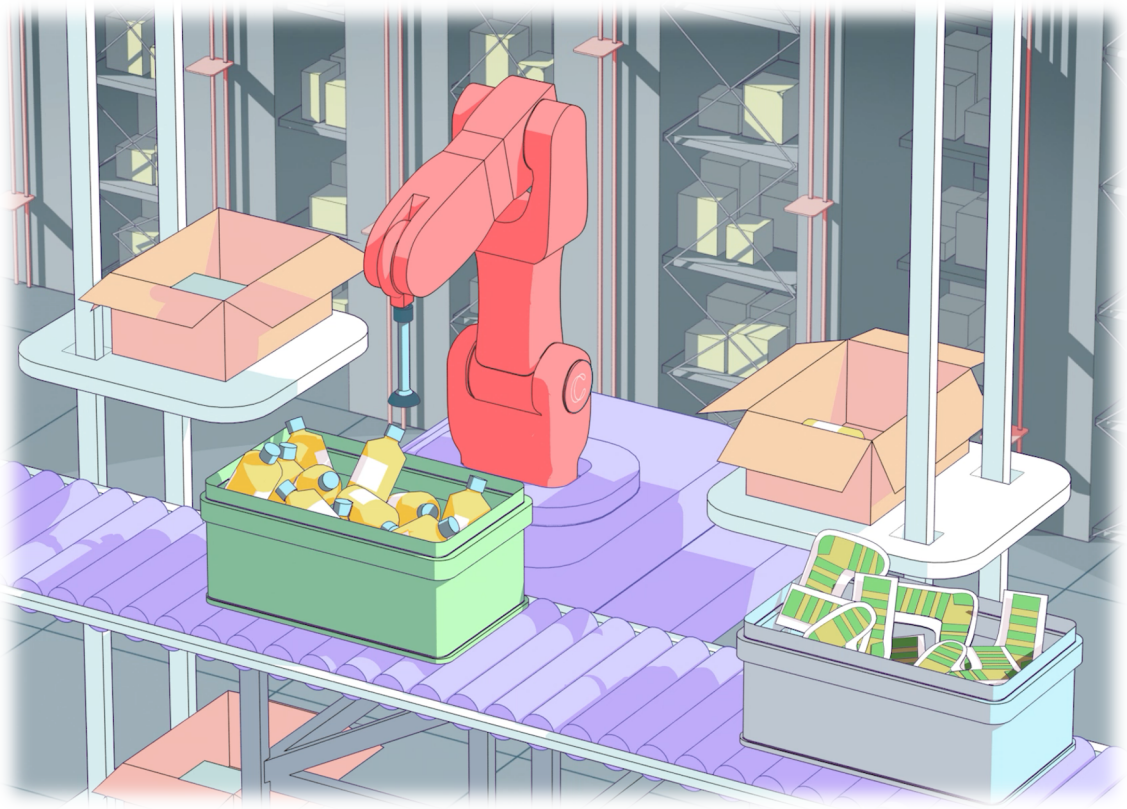
covariant

About us Solutions Our approach Careers Press Blog Contact

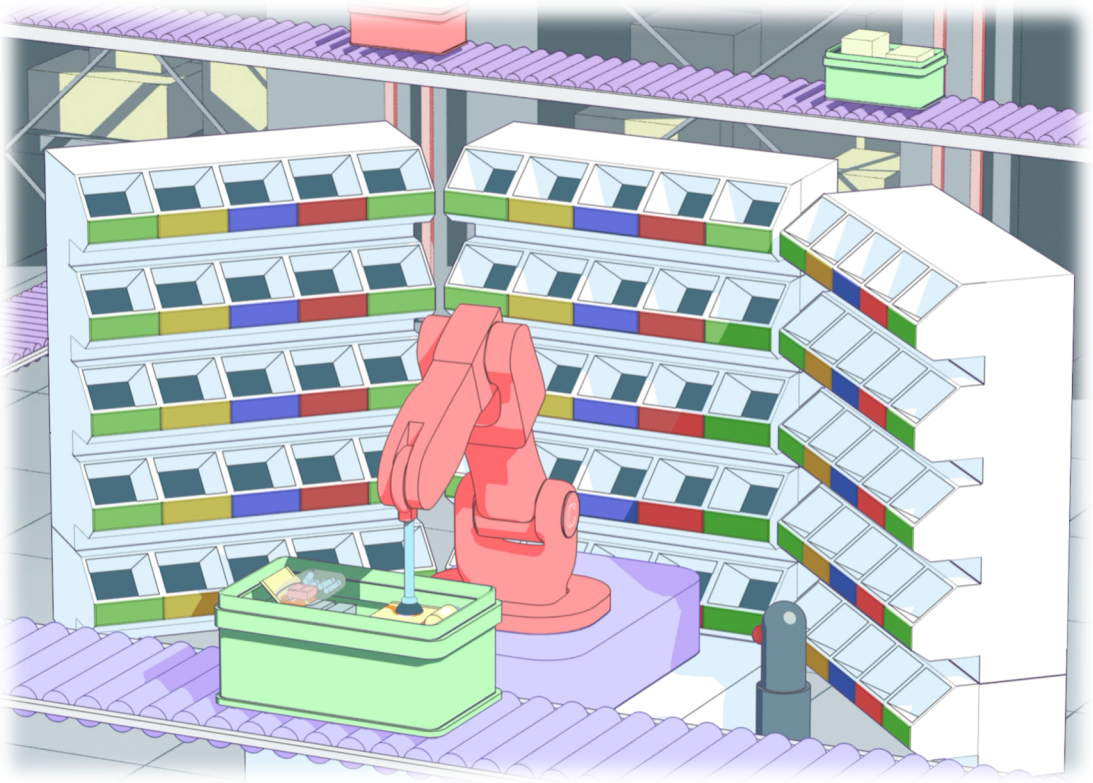

# AI Robotics for the Real World

https://covariant.ai/

# RL in Retail Warehouse or Automotive Factory



Order Picking

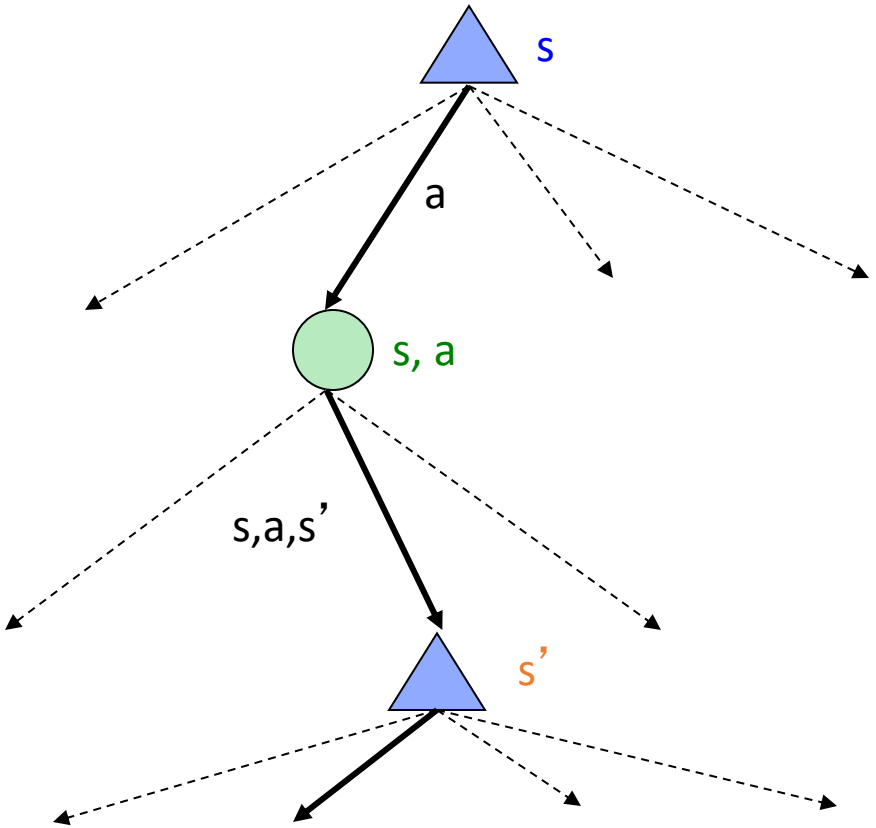


Putwall

https://covariant.ai/solutions

# Markov Decision Processes (MDPs)

- A MDP is defined as:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s, a, s')
    - Probability that a from s leads to s', i.e., P(s'| s, a)
    - Also called the dynamics model
  - A start state
  - Maybe a terminal state
  - A reward function R(s, a, s')
    - reward for the transition (s, a, s')




RL: Unknown Transition Functions or Rewards

# A computational approach to learning from Interactions

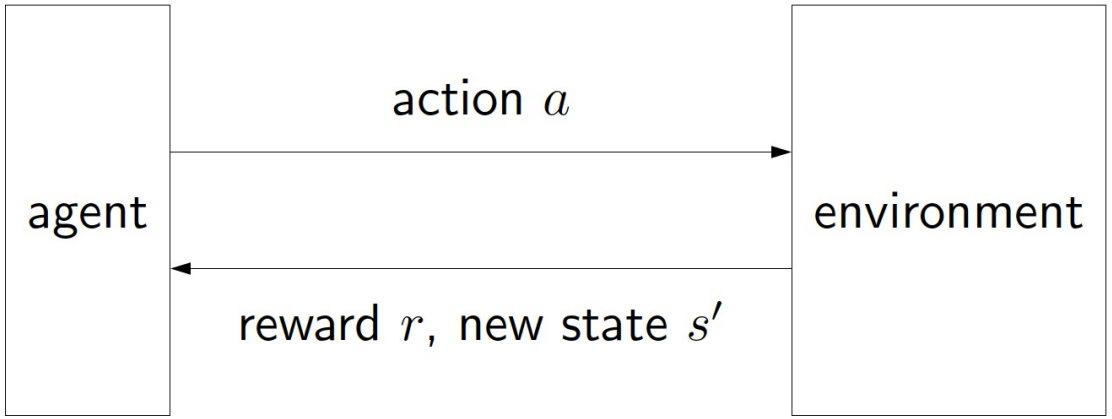
- Reinforcement learning problem:
  - Learning what to do – how to map situations to actions – so as to maximize a numerical reward signal
- Formalize the problem via dynamical systems theory:
  - Markov Decision Processes

# MDPs vs RL

- MDPs (Offline)
  - Known transition model
  - Find a policy to maximize expected utility

- RL (Online)
  - Unknow transition models
  - Perform actions in the world to find out and collect rewards
  - The way humans work: we go through life, taking various actions, getting feedback. We get rewarded for doing well and learn along the way

# Reinforcement Learning Framework




**Algorithm: reinforcement learning template**

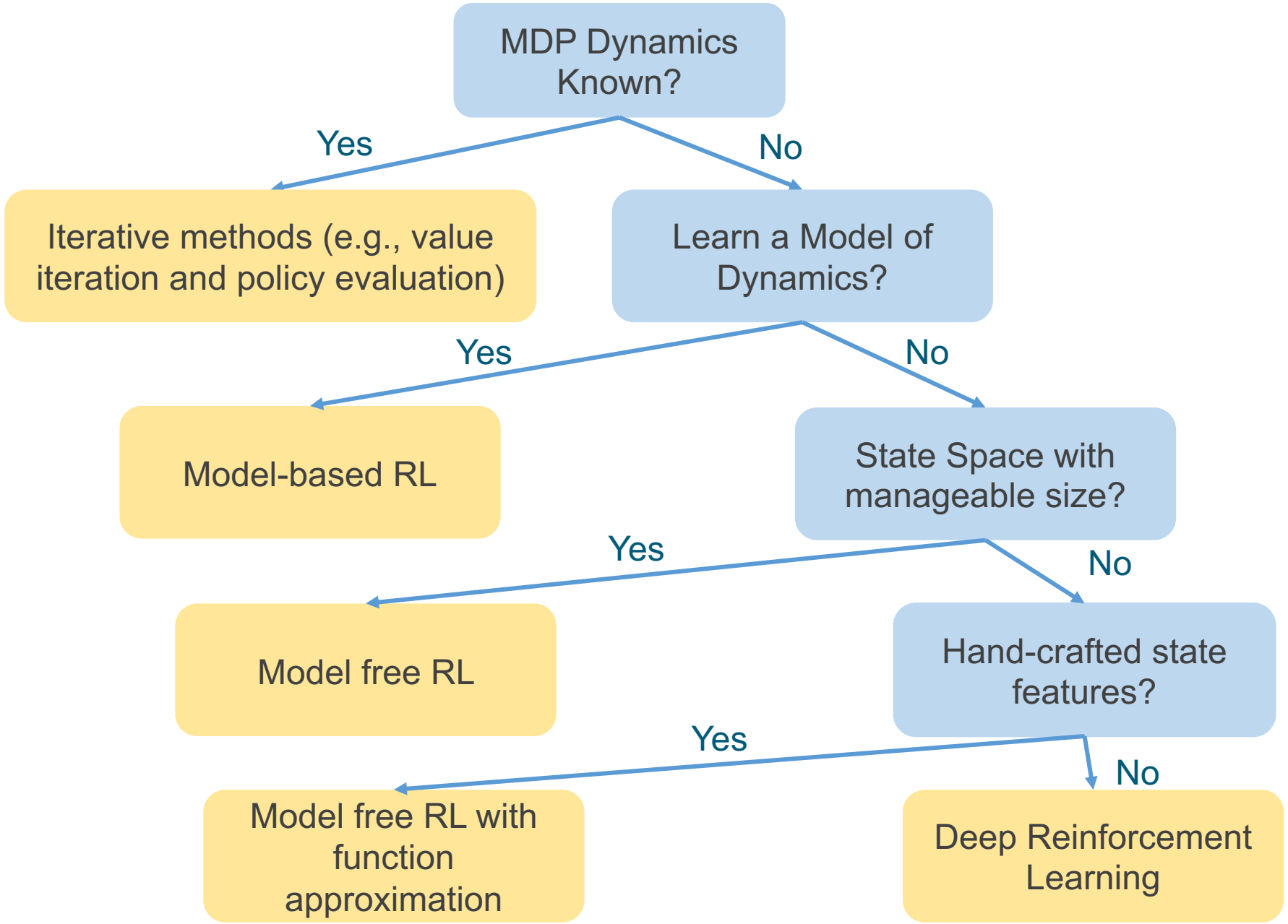
For $t = 1, 2, 3, \ldots$

Choose action $a_t = \pi_{\text{act}}(s_{t-1})$ (**how?**)

Receive reward $r_t$ and observe new state $s_t$

Update parameters (**how?**)

Stanford CS221n


MDP Dynamics Known?
Yes
No
Iterative methods (e.g., value iteration and policy evaluation)
Learn a Model of Dynamics?
Yes
No
Model-based RL
State Space with manageable size?
Yes
No
Model free RL
Hand-crafted state features?
Yes
No
Model free RL with function approximation
Deep Reinforcement Learning

# Model-based RL

- A MDP is defined as:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s, a, s')
    - Probability that a from s leads to s', i.e., P(s'| s, a)
    - Also called the dynamics model
  - A start state
  - Maybe a terminal state
  - A reward function R(s, a, s')
    - reward for the transition (s, a, s')

Estimate transition function and reward in MDP!

# Model-based Monte Carlo

- Monte Carlo is a standard way to estimate the expectation of a random variable by taking an average over samples of that random variable
- Data collected by using policies

$$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \ldots; a_n, r_n, s_n$$

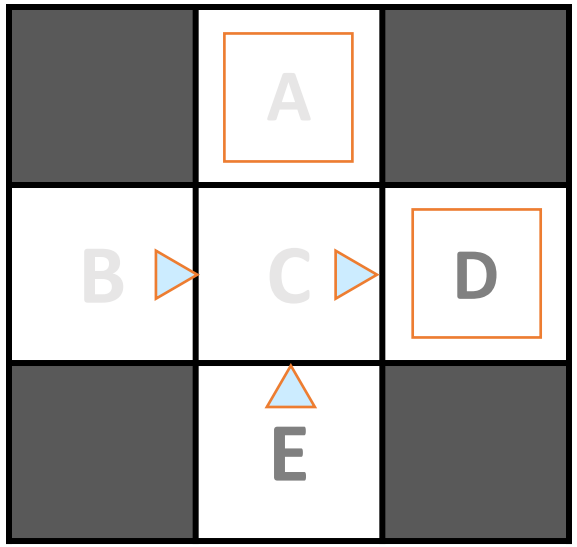
- Transition: $\hat{T}(s, a, s') = \frac{\#\text{ times } (s,a,s') \text{ occurs}}{\#\text{ times } (s,a) \text{ occurs}}$
- Reward: $\widehat{\text{Reward}}(s, a, s') = \text{average of } r \text{ in } (s, a, r, s')$

$$\hat{T}(s,a,s') = \frac{\#\text{ times } (s,a,s') \text{ occurs}}{\#\text{ times } (s,a) \text{ occurs}}$$

$$\widehat{\text{Reward}}(s,a,s') = \text{average of } r \text{ in } (s,a,r,s')$$

# Example

## Input Policy $\pi$




*Assume:* $\gamma = 1$

## Observed Episodes (Training)

### Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

### Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

### Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

### Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

## Learned Model

$\hat{T}(s,a,s')$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
…

$\hat{R}(s,a,s')$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
…

# Model-based Monte Carlo

- Monte Carlo is a standard way to estimate the expectation of a random variable by taking an average over samples of that random variable
- Data collected by using policies

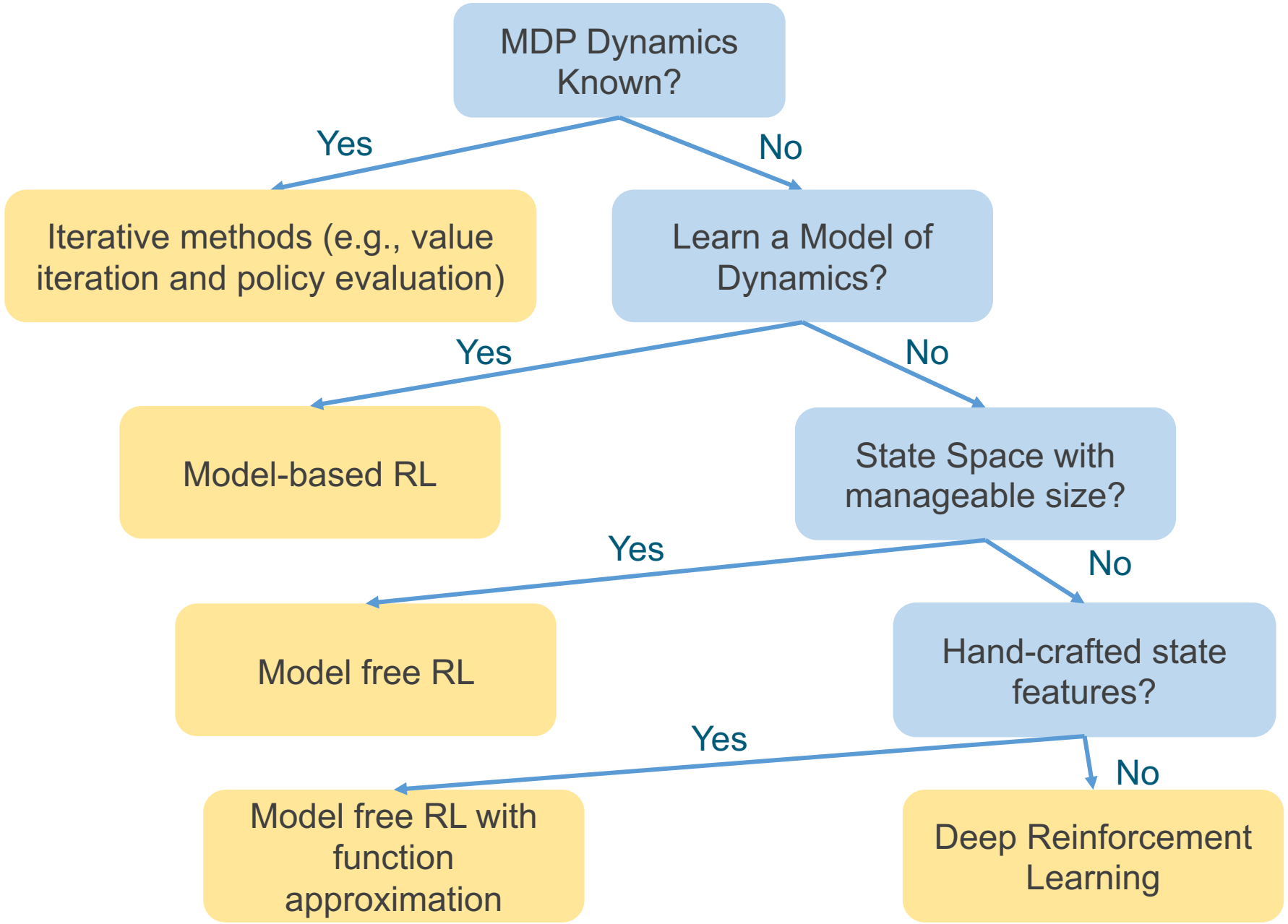
$$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \ldots; a_n, r_n, s_n$$

- Transition: $\hat{T}(s, a, s') = \frac{\#\text{ times } (s,a,s') \text{ occurs}}{\#\text{ times } (s,a) \text{ occurs}}$
- Reward: $\widehat{\text{Reward}}(s, a, s') = \text{average of } r \text{ in } (s, a, r, s')$

Once we know transition and reward, iterative methods can be applied to find optimal policy

# Problem

- If we do not observe a pair (s, a), e.g., (A, ‘south’), we cannot estimate transition function and the corresponding reward
- In reinforcement learning, we need a policy to explore states
  - This is the concept of Exploration!
  - Will discuss later
- This distinguishes reinforcement learning from supervised learning


MDP Dynamics Known?
Yes
No
Iterative methods (e.g., value iteration and policy evaluation)
Learn a Model of Dynamics?
Yes
No
Model-based RL
State Space with manageable size?
Yes
No
Model free RL
Hand-crafted state features?
Yes
No
Model free RL with function approximation
Deep Reinforcement Learning

# Model-free RL

- Estimate optimal value $Q_{opt}(s, a)$ directly
  - we do not explicitly estimate the transitions and rewards
  - Instead, we estimate $Q_{opt}(s, a)$ directly

$$\hat{Q}_{\text{opt}}(s, a) = \sum_{s'} \hat{T}(s, a, s')[\widehat{\text{Reward}}(s, a, s') + \gamma \hat{V}_{\text{opt}}(s')]$$

# Model-free Montel Carlo

- Data (following a policy $\pi$)

$$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \ldots; a_n, r_n, s_n$$

- $Q_\pi(s, a)$ is the expected utility of taking action $a$ from state $s$, and then following policy $\pi$
- Utility:

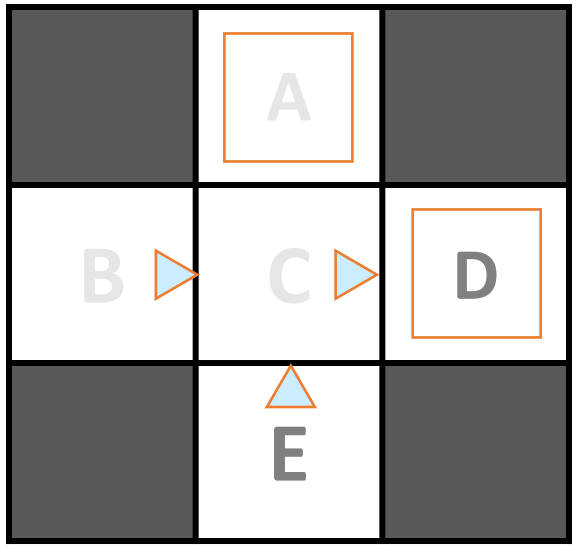
$$U_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

- Estimate:

$$\hat{Q}_\pi(s, a) = \text{average of } u_t \text{ where } s_{t-1} = s, a_t = a$$

# Example

## Input Policy $\pi$




*Assume:* $\gamma = 1$

## Observed Episodes (Training)

### Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

### Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

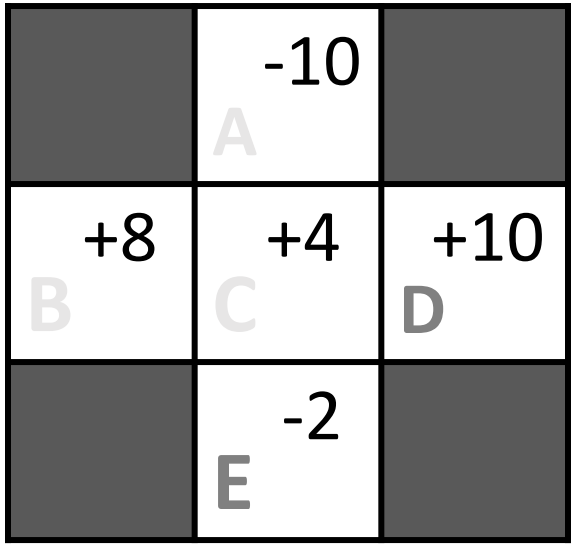
### Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

### Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

## Output Values




Example **C**

E1: (-1 + 10) = 9
E2: (-1 + 10) = 9
E3: (-1 + 10) = 9
E4: (-1 - 10) = -11

Avg: (9+9+9-11)/4 = 4

# On-policy vs off-policy

- Model-based Monte Carlo is off-policy
  - the model does not depend on the exact policy
  - as long as it was able to explore all (s, a) pairs

- Model-free Monte Carlo depends strongly on the policy that is followed
  - the value being computed is dependent on the policy used to generate the data

# A Convex Formulation of Q-value Estimate

- Original Formulation:

$$\hat{Q}_\pi(s,a) = \text{average of } u_t \text{ where } s_{t-1} = s, a_t = a$$

- Averaging as a batch operation that takes a list of numbers and computes the mean
- Iterative procedure for building the mean as new numbers are coming in

On each $(s,a,u)$:

$$\eta = \frac{1}{1+(\#\text{ updates to } (s,a))}$$

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta u$$

On each $(s, a, u)$:

$$\eta = \frac{1}{1+(\#\text{ updates to } (s,a))}$$

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta u$$

Iteration 1: $\eta = \frac{1}{1+0} = 1$

$$\hat{Q}_\pi(s,a) = (1-1) * 0 + 1 * 9 = 9$$

Iteration 2: $\eta = \frac{1}{1+1} = \frac{1}{2}$

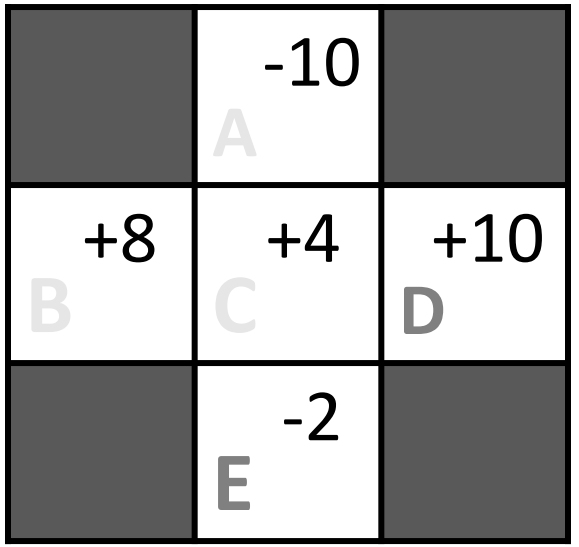
$$\hat{Q}_\pi(s,a) = \left(1 - \frac{1}{2}\right) * 9 + \frac{1}{2} * 9 = \frac{9+9}{2}$$

Iteration 3: $\eta = \frac{1}{1+2} = \frac{1}{3}$

$$\hat{Q}_\pi(s,a) = \left(1 - \frac{1}{3}\right) * \left(\frac{9+9}{2}\right) + \frac{1}{3} * 9 = \frac{9+9+9}{3}$$

Iteration 4: $\eta = \frac{1}{1+3} = \frac{1}{4}$

$$\hat{Q}_\pi(s,a) = \left(1 - \frac{1}{4}\right) * \left(\frac{9+9+9}{3}\right) + \frac{1}{4} * (-11) = \frac{9+9+9+(-11)}{4}$$




Example

**C**

E1: (-1 + 10) = 9

E2: (-1 + 10) = 9

E3: (-1 + 10) = 9

E4: (-1 - 10) = -11

Avg: (9+9+9-11)/4 = 4

# Problem

On each $(s, a, u)$:

$$\eta = \frac{1}{1+(\#\text{ updates to } (s,a))}$$

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta u$$

- We use $u$ as an estimate of $Q_\pi(s,a)$
- $u$ only corresponds to a single episode

$$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \ldots; a_n, r_n, s_n$$

- Could we do better in estimating $Q_\pi(s,a)$?

# SARSA

- A combination of the data and the estimate of $\widehat{Q_\pi}(s,a)$ (based on the data the model has seen before)

**Algorithm: model-free Monte Carlo**

On each $(s,a,u)$:

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta \underbrace{u}_{\text{data}}$$

**Algorithm: SARSA**

On each $(s,a,r,s',a')$:

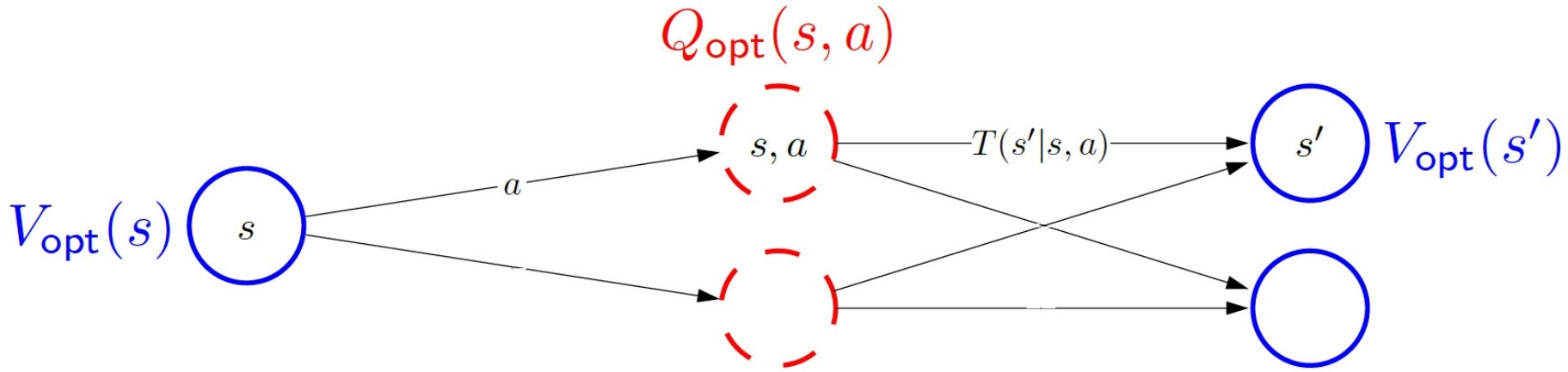
$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta[\underbrace{r}_{\text{data}} + \gamma \underbrace{\hat{Q}_\pi(s',a')}_{\text{estimate}}]$$

# Problem

- Model-free Monte Carlo and SARSA only give us $Q_\pi(s, a)$
- How to obtain $Q_{opt}(s, a)$ to act optimally?

| **Output** | **MDP** | **reinforcement learning** |
|---|---|---|
| $Q_\pi$ | policy evaluation | model-free Monte Carlo, SARSA |
| $Q_{\text{opt}}$ | value iteration | **Q-learning** |

# Optimal Q-value




Optimal Q-value if take action $a$ in state $s$

$$Q_{\text{opt}}(s,a) = \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}(s')].$$

How to obtain optimal Q-value in a model-free manner?

# Q-Learning

**Algorithm: SARSA**

On each $(s, a, r, s', a')$:

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta[\underbrace{r}_{\text{data}} + \gamma \underbrace{\hat{Q}_\pi(s',a')}_{\text{estimate}}]$$

Optimal Q-value: $Q_{\text{opt}}(s,a) = \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}(s')]$.

**Algorithm: Q-learning [Watkins/Dayan, 1992]**

On each $(s, a, r, s')$:

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\hat{Q}_{\text{opt}}(s,a) + \eta(r + \gamma \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s',a'))]$$

- We don't have an expectation over $s'$, but only have one sample $s'$
- We do not replace $\widehat{Q_\pi}(s,a)$ with the target value, but to interpolate between the old value (prediction) and the new value (target)
- The difference between Q-learning and SARSA is that $\widehat{V_{opt}}(s')$ involves a maximum over actions rather than just taking the action of the policy

Stanford CS221n

## Algorithm: SARSA

On each $(s, a, r, s', a')$:

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta(r + \gamma \hat{Q}_\pi(s',a'))$$

## Algorithm: Q-learning [Watkins/Dayan, 1992]

On each $(s, a, r, s')$:

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\hat{Q}_{\text{opt}}(s,a) + \eta(r + \gamma \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s',a'))]$$

Initialized

| Q-Table | | Actions | | | | | |
|---|---|---|---|---|---|---|---|
| | | South (0) | North (1) | East (2) | West (3) | Pickup (4) | Dropoff (5) |
| States | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | 327 | 0 | 0 | 0 | 0 | 0 | 0 |
| | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | 499 | 0 | 0 | 0 | 0 | 0 | 0 |

Training

| Q-Table | | Actions | | | | | |
|---|---|---|---|---|---|---|---|
| | | South (0) | North (1) | East (2) | West (3) | Pickup (4) | Dropoff (5) |
| States | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | 328 | -2.30108105 | -1.97092096 | -2.30357004 | -2.20591839 | -10.3607344 | -8.5583017 |
| | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | 499 | 9.96984239 | 4.02706992 | 12.96022777 | 29 | 3.32877873 | 3.38230603 |

Q–Learning table of states by actions that is initialized to zero, then each cell is updated through training.

# How to implement Q-learning?

https://en.wikipedia.org/wiki/Q-learning

# Is Q-learning on-policy or off-policy?

**Algorithm: SARSA**

On each $(s, a, r, s', a')$:

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta(r + \gamma \hat{Q}_\pi(s',a'))$$

**Algorithm: Q-learning [Watkins/Dayan, 1992]**

On each $(s, a, r, s')$:

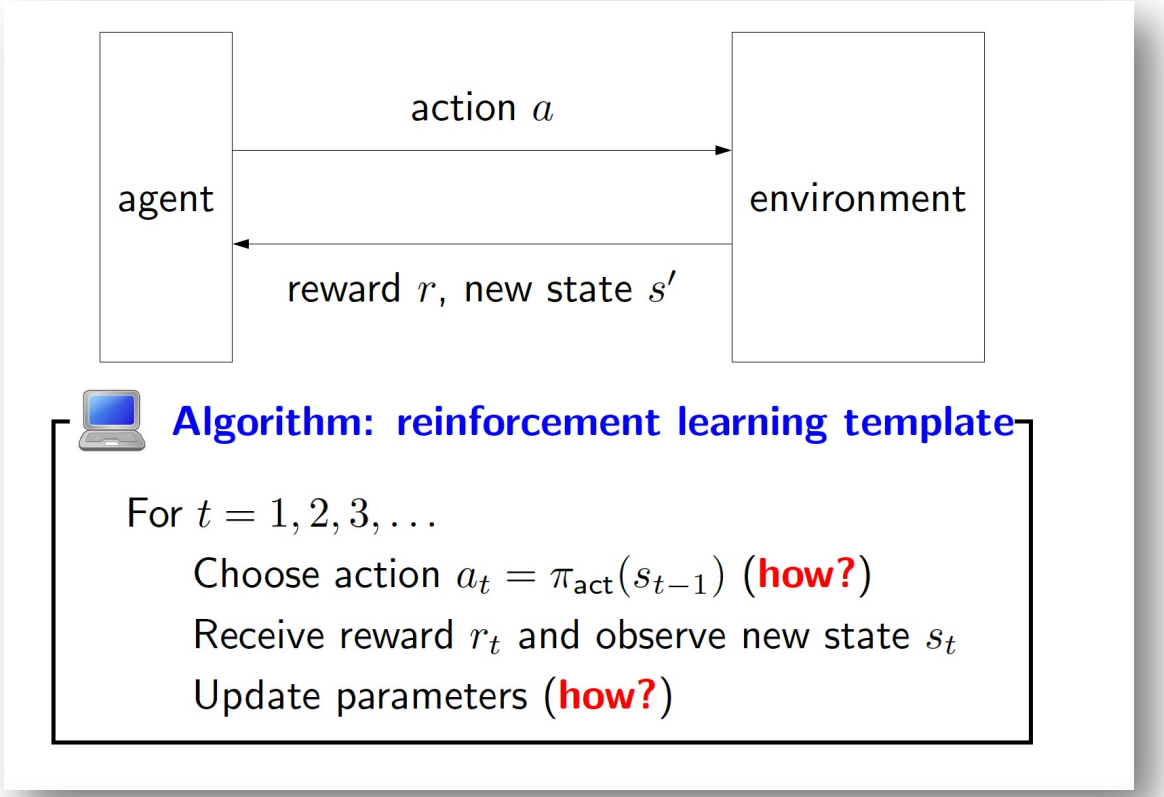
$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\hat{Q}_{\text{opt}}(s,a) + \eta(r + \gamma \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s',a'))]$$

Model-free Q-learning is off-policy, since it can learn the optimal policy using data from other policies

# Different RL Algorithms

| **Algorithm** | **Estimating** | **Based on** |
|---|---|---|
| Model-Based Monte Carlo | $\hat{T}, \hat{R}$ | $s_0, a_1, r_1, s_1, \ldots$ |
| Model-Free Monte Carlo | $\hat{Q}_\pi$ | $u$ |
| SARSA | $\hat{Q}_\pi$ | $r + \hat{Q}_\pi$ |
| Q-Learning | $\hat{Q}_{\text{opt}}$ | $r + \hat{Q}_{\text{opt}}$ |

# How to determine the exploration policy?




Determine an exploration policy to act to collect data for learning

# Exploration Policy

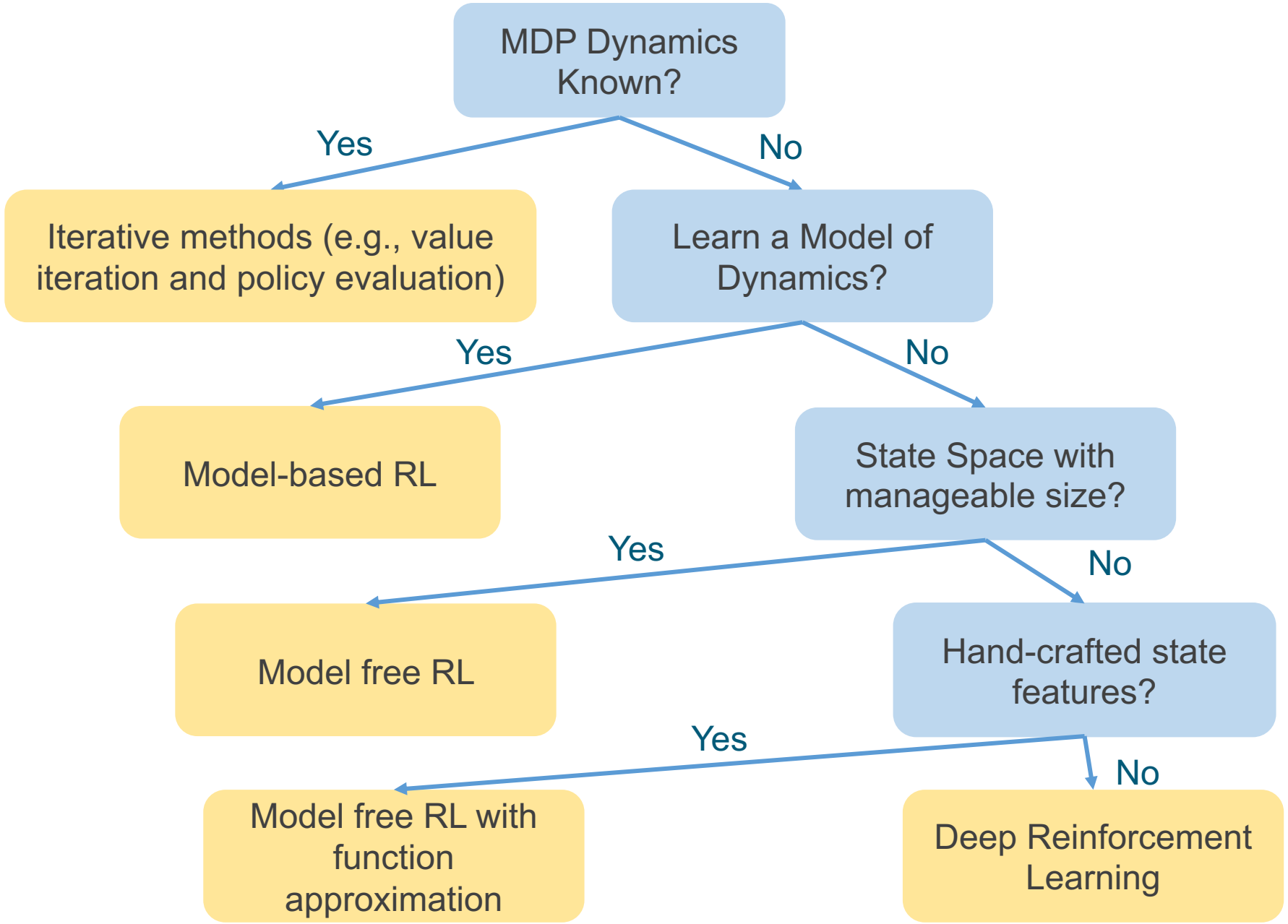
- No Exploration, All Exploitation
  - Set $\pi_{act} = \arg\max_{a \in Actions(s)} \widehat{Q_\pi}(s, a)$
  - Once the agent finds an optimal policy, they will not try other actions
- No Exploitation, all exploration
  - Set $\pi_{act} =$ random from Actions (s)
  - it doesn't exploit what it learns and ends up with a very low utility

# Epsilon-greedy Algorithm

- The natural thing to do when you have two extremes is to interpolate between the two
- Start with high

**Algorithm: epsilon-greedy policy**

$$\pi_{\text{act}}(s) = \begin{cases} \arg\max_{a \in \text{Actions}} \hat{Q}_{\text{opt}}(s,a) & \text{probability } 1-\epsilon, \\ \text{random from Actions}(s) & \text{probability } \epsilon. \end{cases}$$


MDP Dynamics Known?
Yes
No
Iterative methods (e.g., value iteration and policy evaluation)
Learn a Model of Dynamics?
Yes
No
Model-based RL
State Space with manageable size?
Yes
No
Model free RL
Hand-crafted state features?
Yes
No
Model free RL with function approximation
Deep Reinforcement Learning

# In Q-learning

- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow \hat{Q}_{\text{opt}}(s,a) - \eta[\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} - \underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}]$$

# Function Approximation

Similar to the evaluation function discussed in the Adversarial Search

- Function approximation fixes the issue of large lookup tables by parameterizing $Q_{opt}(s, a)$ by a weight vector and a feature vector

**Key idea: linear regression model**

Define **features** $\phi(s, a)$ and **weights** $\mathbf{w}$:

$$\hat{Q}_{\text{opt}}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$

Do you remember where we learn this?

# Q-learning with Function Approximation

- We hope $\hat{Q}_{opt}(s,a;\boldsymbol{w}) \approx Q_{opt}(s,a)$ by using some parametric functions
- Loss function: MSE

$$\arg\min_{\boldsymbol{w}}\left[\hat{Q}_{opt}(s,a;\boldsymbol{w}) - Q_{opt}(s,a)\right]^2$$

$$= \arg\min_{\boldsymbol{w}}\left[\hat{Q}_{opt}(s,a;\boldsymbol{w}) - (r + \gamma\hat{V}_{opt}(s'))\right]^2$$

- Gradient descent (partial derivative with respect to $\boldsymbol{w}$)

Objective Function

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\left[\hat{Q}_{opt}(s,a;\boldsymbol{w}) - (r + \gamma\hat{V}_{opt}(s'))\right]\cdot \hat{Q}_{opt}(s,a;\boldsymbol{w})$$

prediction target

# Gradient Decent

- Objective function:

$$TrainLoss(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} (\theta\phi(x) - y)^2$$

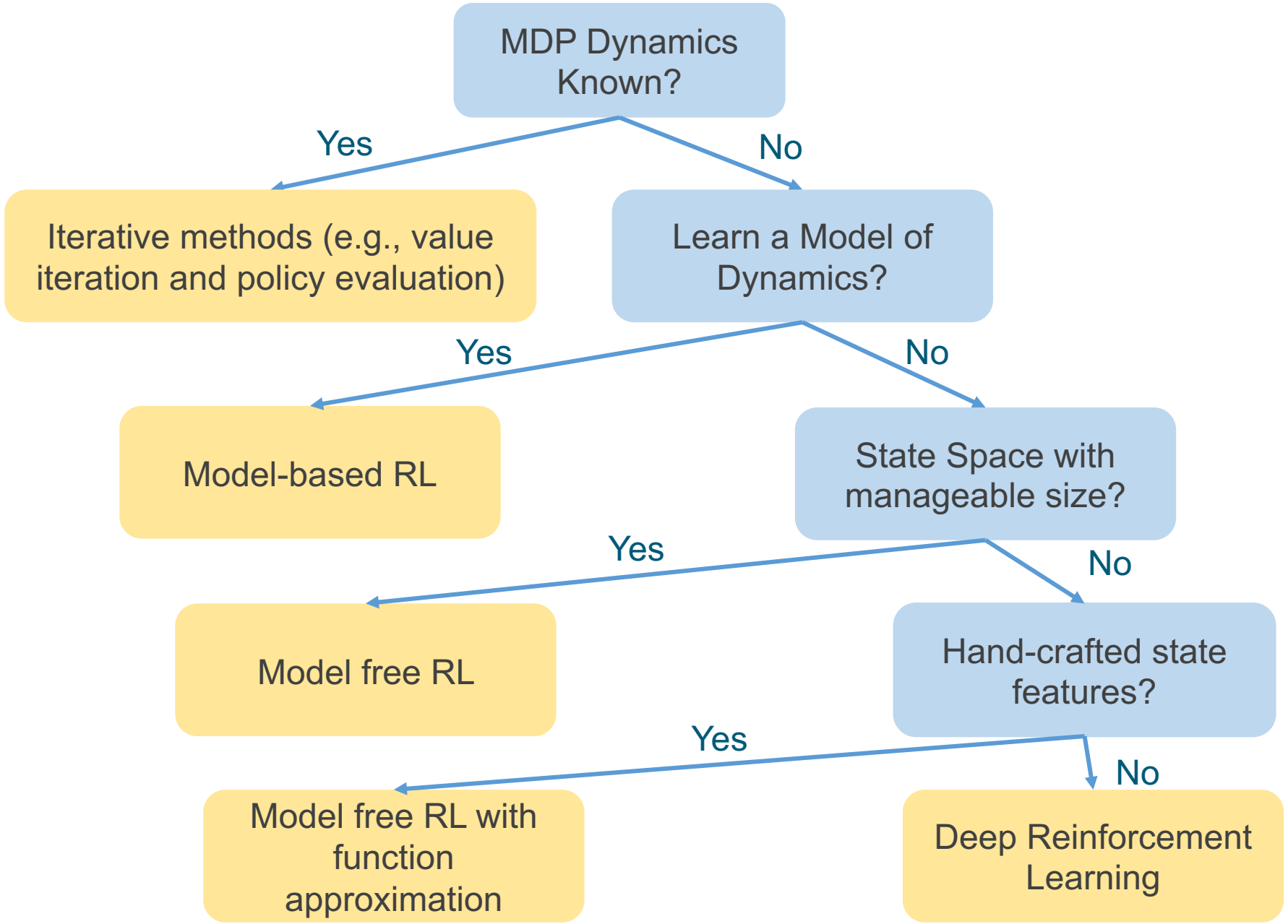
- Partial derivative with respect to $\theta$ :

$$\nabla_\theta TrainLoss(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} 2(\theta\phi(x) - y)\,\phi(x)$$

- Update:

$$\theta \leftarrow \theta - \eta \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} 2(\theta\phi(x) - y)\,\phi(x)$$

Stanford CS231n Note


MDP Dynamics Known?
Yes
No
Iterative methods (e.g., value iteration and policy evaluation)
Learn a Model of Dynamics?
Yes
No
Model-based RL
State Space with manageable size?
Yes
No
Model free RL
Hand-crafted state features?
Yes
No
Model free RL with function approximation
Deep Reinforcement Learning

# Deep Reinforcement Learning

- Instead of manually determine the states, let's take raw data (i.e., images) as states
- Parameterize images and $\hat{Q}_{opt}(s, a; \boldsymbol{w})$ with neural networks
  - Deep Q-Networks (Deepmind, 2015)
- The field of Deep RL is growing very fast!
  - http://rail.eecs.berkeley.edu/deeprlcourse/

# Three Different Learning Paradigms

- Supervised Learning
  - the model learns from training data that consists of a labeled pair of inputs and outputs
  - For example, in Chess, you have to all the moves a player can make in each state, along with labels indicating whether it is a good move or not
- Unsupervised learning
  - the model learns the hidden structure in the input data
- Reinforcement Learning
  - the model learns by **maximizing the cumulative reward**
  - the data is the **Environment that you are interacting with**