ESSENTIALS OF RL

Reinforcement Learning: Lecture 2

3rd Nepal Winter School in AI

24th Dec 2021

Abhishek Naik

OUTLINE

- Dynamic Programming (DP)
- Temporal-Difference (TD) Learning
- Model-based RL
- Policy Optimization

REINFORCEMENT LEARNING

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$\pi_0 \longrightarrow \pi^*$$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

π_0 π^*

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

π_0 π_1 π^*

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

π_0 π_1 π_2 π^*

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$\pi_0 \quad \pi_1 \quad \pi_2 \quad \pi_{n-1} \quad \pi^*$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$\pi_0 \qquad \pi_1 \qquad \pi_2 \qquad \qquad \pi_{n-1} \qquad \pi^*$$

$$\pi_0 \rightarrow v_{\pi_0}$$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$
\begin{tikzcd}[column sep=small]
\pi_0 \arrow[dr] & & \pi_1 & \pi_2 & \cdots & \pi_{n-1} & \pi^* \\
& v_{\pi_0} \arrow[ur] & & & & &
\end{tikzcd}
$$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$
\begin{tikzcd}
\pi_0 \arrow[dr] & & \pi_1 \arrow[dr] & & \pi_2 & & \pi_{n-1} & & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} & & & & &
\end{tikzcd}
$$

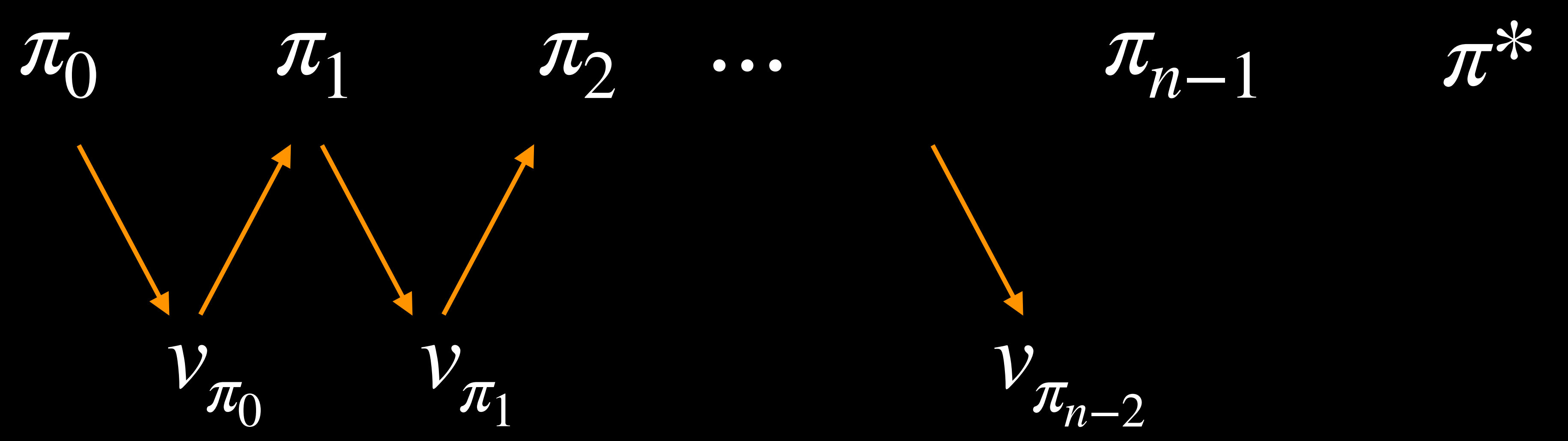
REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$
\begin{tikzcd}
\pi_0 \arrow[dr] & & \pi_1 \arrow[dr] & & \pi_2 & \cdots & & \pi_{n-1} & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} \arrow[ur] & & & & &
\end{tikzcd}
$$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*



REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$
\begin{tikzcd}[column sep=small]
\pi_0 \arrow[dr] & & \pi_1 \arrow[dr] & & \pi_2 & \cdots & {} \arrow[dr] & & \pi_{n-1} & & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} \arrow[ur] & & & & v_{\pi_{n-2}} \arrow[ur] & & &
\end{tikzcd}
$$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$
\begin{tikzcd}
\pi_0 \arrow[dr] & & \pi_1 \arrow[dr] & & \pi_2 & \cdots \arrow[dr] & & \pi_{n-1} \arrow[dr] & & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} \arrow[ur] & & & v_{\pi_{n-2}} \arrow[ur] & & v_{\pi_{n-1}} &
\end{tikzcd}
$$

REINFORCEMENT LEARNING

- Goal: learning some behaviour to maximize a numerical reward signal
  - via *trial and error*
  - with potentially *delayed rewards.*

$$
\begin{tikzcd}
\pi_0 \arrow[dr] & & \pi_1 \arrow[dr] & & \pi_2 & \cdots & {} \arrow[dr] & & \pi_{n-1} \arrow[dr] & & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} \arrow[ur] & & & & v_{\pi_{n-2}} \arrow[ur] & & v_{\pi_{n-1}} \arrow[ur] &
\end{tikzcd}
$$

REINFORCEMENT LEARNING

REINFORCEMENT LEARNING

- Sub-goal: evaluate a policy

REINFORCEMENT LEARNING

- Sub-goal: evaluate a policy
  - estimate the value function for a given policy

REINFORCEMENT LEARNING

- Sub-goal: evaluate a policy
  - estimate the value function for a given policy

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

REINFORCEMENT LEARNING

- Sub-goal: evaluate a policy
  - estimate the value function for a given policy

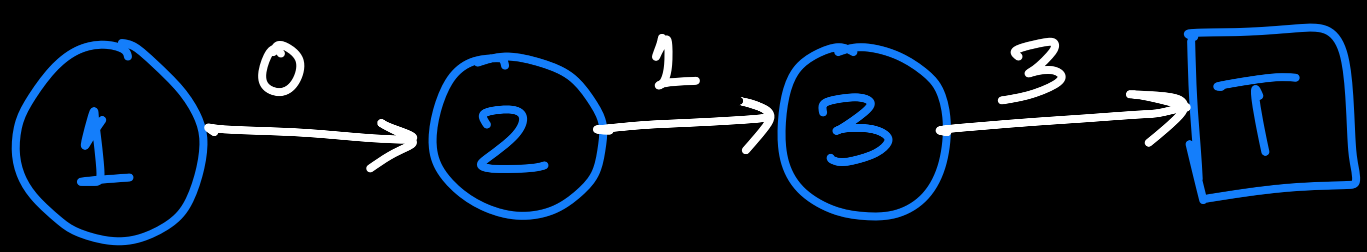
$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

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

REINFORCEMENT LEARNING

- Sub-goal: evaluate a policy
  - estimate the value function for a given policy

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \qquad G_t = R_{t+1} + R_{t+2} + \ldots + R_T$$

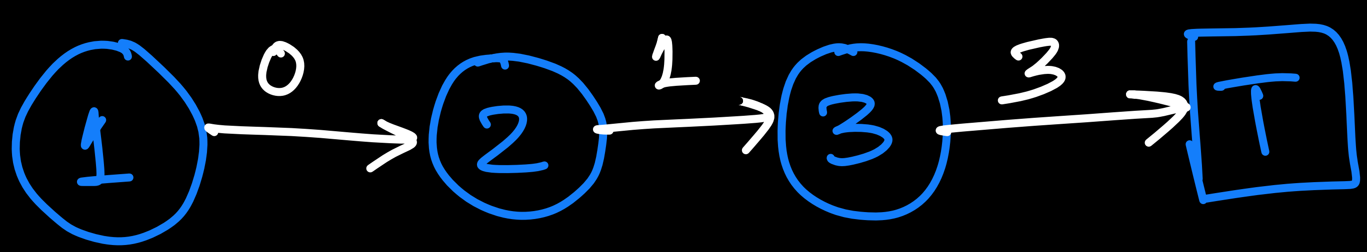


REINFORCEMENT LEARNING

- Sub-goal: evaluate a policy
  - estimate the value function for a given policy

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \qquad G_t = R_{t+1} + R_{t+2} + \ldots + R_T$$

$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$



FUNDAMENTALS

$$v_\pi(s) = \sum_a \pi(a \,|\, s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + v_\pi(s')\big]$$

FUNDAMENTALS

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

Bellman equation

FUNDAMENTALS

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

Bellman equation

Evaluating the optimal policy, π^*:

FUNDAMENTALS

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

Bellman equation

Evaluating the optimal policy, π^*:

$$v_{\pi^*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi^*}(s')\right]$$

FUNDAMENTALS

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\big[r + v_\pi(s')\big]$$

Bellman equation

Evaluating the optimal policy, π^*:

$$v_{\pi^*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + v_{\pi^*}(s')\big]$$

Bellman optimality equation

FUNDAMENTALS

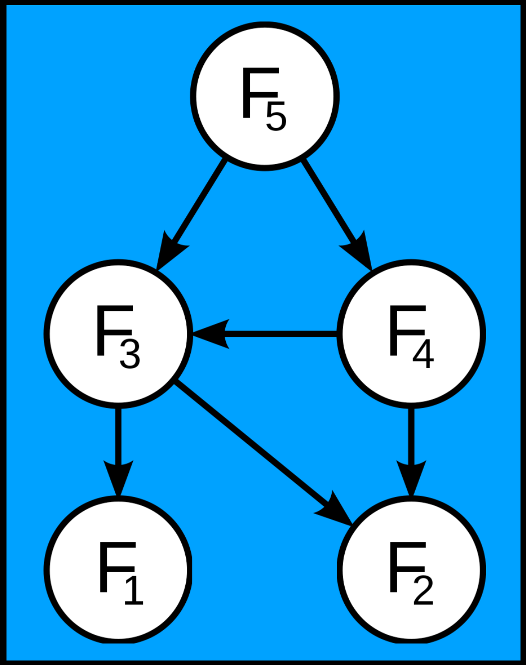
$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

Bellman equation

Evaluating the optimal policy, π^*:

$$v_{\pi^*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi^*}(s')\right]$$

Bellman optimality equation



FUNDAMENTALS

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$
$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

$$v_\pi(s) = \sum_a \pi(a \mid s)\, q_\pi(s,a)$$

$$v_{\pi*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi*}(s')\right]$$

$$q_{\pi*}(s,a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_{\pi*}(s', a')\right]$$

FUNDAMENTALS

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

$$v_\pi(s) = \sum_a \pi(a \mid s)\, q_\pi(s, a)$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right] = \sum_{s',r} p(s', r \mid s, a)\left[r + \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

$$v_{\pi*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi*}(s')\right]$$

$$q_{\pi*}(s, a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_{\pi*}(s', a')\right]$$

FUNDAMENTALS

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$
$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

$$v_\pi(s) = \sum_a \pi(a \mid s)\, q_\pi(s, a)$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

$$v_{\pi*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi*}(s')\right]$$

$$q_{\pi*}(s, a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_{\pi*}(s', a')\right]$$

FUNDAMENTALS

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$
$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

$$v_\pi(s) = \sum_a \pi(a \mid s)\, q_\pi(s, a)$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

$$v_{\pi*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi*}(s')\right]$$

$$q_{\pi*}(s, a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_{\pi*}(s', a')\right]$$

FUNDAMENTALS

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$
$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

$$v_\pi(s) = \sum_a \pi(a \mid s)\, q_\pi(s, a)$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

$$v_{\pi*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi*}(s')\right]$$

$$q_{\pi*}(s, a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_{\pi*}(s', a')\right]$$

FUNDAMENTALS

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$
$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + v_\pi(s')\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

$$v_\pi(s) = \sum_a \pi(a \mid s)\, q_\pi(s, a)$$

$$v_{\pi*}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + v_{\pi*}(s')\right]$$

$$q_{\pi*}(s,a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_{\pi*}(s', a')\right]$$

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,
  - *iteratively* improve estimates

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,
  - *iteratively* improve estimates
  - with the help of *bootstrapping.*

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,
  - *iteratively* improve estimates
  - with the help of *bootstrapping*.

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + v_t(s') \right]$$

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,
  - *iteratively* improve estimates
  - with the help of *bootstrapping*.

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + v_t(s') \right]$$

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,
  - *iteratively* improve estimates
  - with the help of *bootstrapping*.

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + v_t(s') \right]$$

CORE IDEAS OF DYNAMIC PROGRAMMING (DP)

- Goal:
  - evaluate a given policy (the *prediction* problem), or
  - find the best policy (the *control* problem).
- Methodology:
  - Given a *model* of the world,
  - *iteratively* improve estimates
  - with the help of *bootstrapping*.

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + v_t(s') \right]$$

WE DO NOT USUALLY HAVE A MODEL OF THE WORLD

WE DO NOT USUALLY HAVE A MODEL OF THE WORLD

- But we can (and do!) interact with the world.

WE DO NOT USUALLY HAVE A MODEL OF THE WORLD

- But we can (and do!) interact with the world.
- Can we use that experience to improve our estimates?

WE DO NOT USUALLY HAVE A MODEL OF THE WORLD

- But we can (and do!) interact with the world.
- Can we use that experience to improve our estimates?

$$v_{t+1}(s) \to \left[r + v_t(s')\right]$$

WE DO NOT USUALLY HAVE A MODEL OF THE WORLD

- But we can (and do!) interact with the world.
- Can we use that experience to improve our estimates?

$$v_{t+1}(s) \rightarrow \left[r + v_t(s')\right]$$

How can we improve estimates
from a *stream* of data?

WE DO NOT USUALLY HAVE A MODEL OF THE WORLD

- But we can (and do!) interact with the world.
- Can we use that experience to improve our estimates?

$$v_{t+1}(s) \to \left[r + v_t(s')\right]$$

How can we improve estimates
from a *stream* of data?

$$S_0, A_0, R_1, S_1, \ldots, S_t, A_t, R_{t+1}, S_{t+1}, \ldots$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

$$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \cdots$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

$$\bar{x}_{N+1} = \frac{1}{N+1}(\sum_{i=1}^{N} x_i + x_{N+1})$$

- An example for intuition:
  - Compute the average of a stream of samples

$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \ldots$

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

$$\bar{x}_{N+1} = \frac{1}{N+1}\left(\sum_{i=1}^{N} x_i + x_{N+1}\right)$$

$$= \frac{N}{N+1}\frac{1}{N}\sum_{i=1}^{N} x_i + \frac{1}{N+1}x_{N+1}$$

$$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \ldots$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

$$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \ldots$$

$$\begin{aligned}
\bar{x}_{N+1} &= \frac{1}{N+1}\left(\sum_{i=1}^{N} x_i + x_{N+1}\right) \\
&= \frac{N}{N+1}\frac{1}{N}\sum_{i=1}^{N} x_i + \frac{1}{N+1}x_{N+1} \\
&= \frac{N}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1}
\end{aligned}$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

$$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \ldots$$

$$\begin{aligned}
\bar{x}_{N+1} &= \frac{1}{N+1}\left(\sum_{i=1}^{N} x_i + x_{N+1}\right) \\
&= \frac{N}{N+1}\frac{1}{N}\sum_{i=1}^{N} x_i + \frac{1}{N+1}x_{N+1} \\
&= \frac{N}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1} \\
&= \left(1 - \frac{1}{N+1}\right)\bar{x}_N + \frac{1}{N+1}x_{N+1}
\end{aligned}$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

$$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \ldots$$

$$\begin{aligned}
\bar{x}_{N+1} &= \frac{1}{N+1}\left(\sum_{i=1}^{N} x_i + x_{N+1}\right) \\
&= \frac{N}{N+1}\frac{1}{N}\sum_{i=1}^{N} x_i + \frac{1}{N+1}x_{N+1} \\
&= \frac{N}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1} \\
&= \left(1 - \frac{1}{N+1}\right)\bar{x}_N + \frac{1}{N+1}x_{N+1} \\
&= \bar{x}_N + \frac{1}{N+1}(x_{N+1} - \bar{x}_N)
\end{aligned}$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

- An example for intuition:
  - Compute the average of a stream of samples

$x_1, x_2, x_3, \ldots, x_N, x_{N+1}, \ldots$

$$
\begin{aligned}
\bar{x}_{N+1} &= \frac{1}{N+1}\left(\sum_{i=1}^{N} x_i + x_{N+1}\right) \\
&= \frac{N}{N+1}\frac{1}{N}\sum_{i=1}^{N} x_i + \frac{1}{N+1}x_{N+1} \\
&= \frac{N}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1} \\
&= \left(1 - \frac{1}{N+1}\right)\bar{x}_N + \frac{1}{N+1}x_{N+1} \\
&= \bar{x}_N + \frac{1}{N+1}(x_{N+1} - \bar{x}_N) \\
\bar{x}_{N+1} &= \bar{x}_N + \alpha(x_{N+1} - \bar{x}_N)
\end{aligned}
$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

$$\bar{x}_{N+1} = \bar{x}_N + \alpha\left(x_{N+1} - \bar{x}_N\right)$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

$$\bar{x}_{N+1} = \bar{x}_N + \alpha\left(x_{N+1} - \bar{x}_N\right)$$

new_estimate = old_estimate + stepsize * (new_target - old_estimate)

IMPROVING ESTIMATES FROM A STREAM OF DATA

$$\bar{x}_{N+1} = \bar{x}_N + \alpha \left(x_{N+1} - \bar{x}_N \right)$$

new_estimate = old_estimate + stepsize * (new_target - old_estimate)

DP $$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + v_t(s') \right]$$

IMPROVING ESTIMATES FROM A STREAM OF DATA

$$\bar{x}_{N+1} = \bar{x}_N + \alpha \left(x_{N+1} - \bar{x}_N\right)$$

new_estimate = old_estimate + stepsize * (new_target - old_estimate)

DP $$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + v_t(s')\right]$$

TD $$v_{t+1}(s) = v_t(s) + \alpha\left[\left(r + v_t(s')\right) - v_t(s)\right]$$

TEMPORAL-DIFFERENCE (TD) LEARNING

TEMPORAL-DIFFERENCE (TD) LEARNING

Algorithm : Tabular TD learning to estimate v_π

Input: The target policy π

Algorithm parameters: step size $\alpha \in (0, 1]$

Initialize $V(s)$, for all $s \in \mathcal{S}$, arbitrarily (e.g., to zero)

Observe initial state S

for *each time step* **do**

  $A \leftarrow$ action according to π in S

  Take action A, observe R, S'

  $V(S) \leftarrow V(S) + \alpha \left[R + V(S') - V(S)\right]$

  $S \leftarrow S'$

end

return V

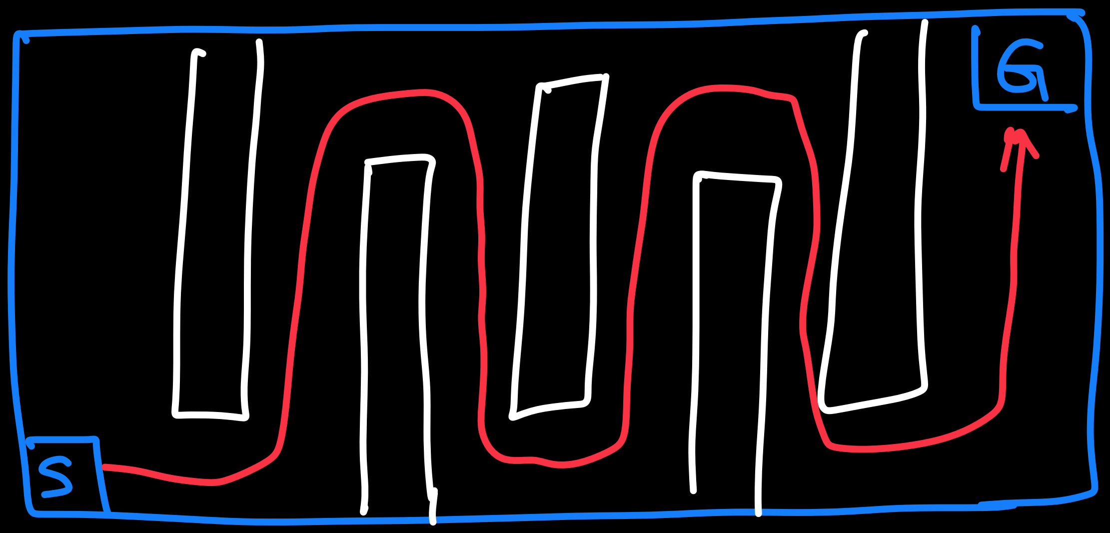
CONTROL: EXPLORATION VS EXPLOITATION

CONTROL: EXPLORATION VS EXPLOITATION

- Simple heuristic:
  - with a small probability, pick a random action

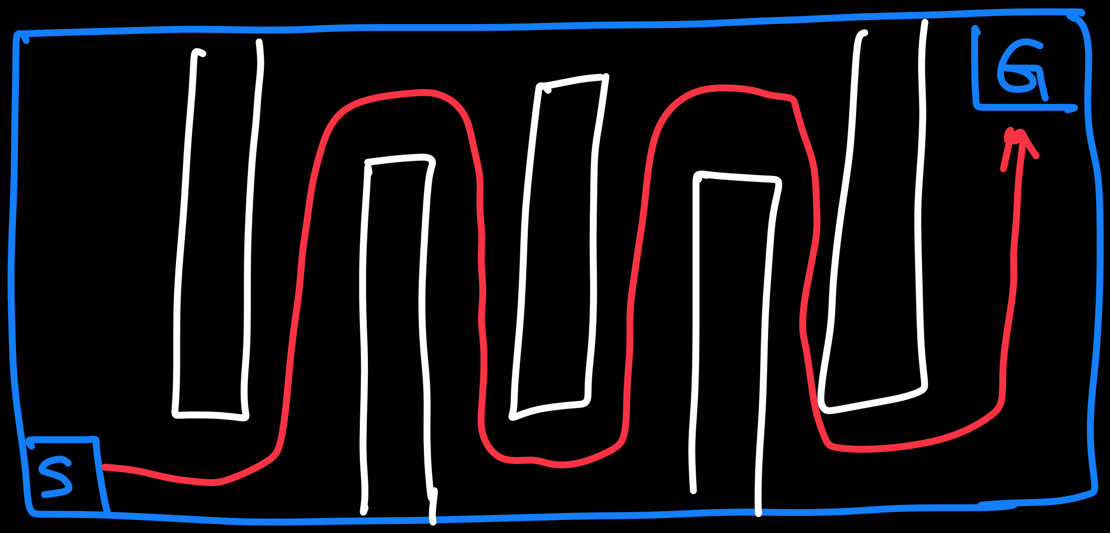
CONTROL: EXPLORATION VS EXPLOITATION

- Simple heuristic:
  - with a small probability, pick a random action



CONTROL: EXPLORATION VS EXPLOITATION

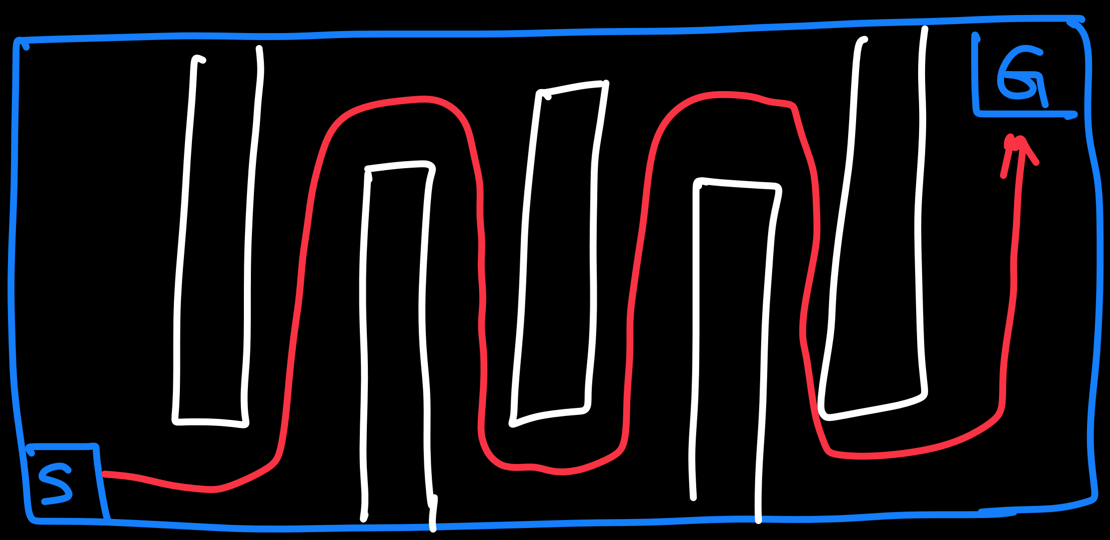
- Simple heuristic:
  - with a small probability, pick a random action



- With probability ϵ, pick an action randomly
- With probability 1-ϵ, pick the 'greedy' action

CONTROL: EXPLORATION VS EXPLOITATION

- Simple heuristic:
  - with a small probability, pick a random action



- With probability ϵ, pick an action randomly
- With probability 1-ϵ, pick the 'greedy' action

ϵ-greedy action selection

CONTROL ALGORITHM: SARSA

Algorithm : SARSA to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, arbitrarily (e.g., to zero)
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $A' \leftarrow$ action in S' according to policy derived from Q (e.g., ϵ-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + Q(S', A') - Q(S, A)\right]$
    $S \leftarrow S'$
end

CONTROL ALGORITHM: SARSA

Algorithm : SARSA to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily (e.g., to zero)
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $A' \leftarrow$ action in S' according to policy derived from Q (e.g., ϵ-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + Q(S', A') - Q(S, A)\right]$
    $S \leftarrow S'$
end

"On-policy"

CONTROL ALGORITHM: Q-LEARNING

Algorithm : Q-learning to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, arbitrarily (e.g., to zero)

Observe initial state S

for *each time step* **do**

  $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)

  Take action A, observe R, S'

  $A' \leftarrow$ action in S' according to the greedy policy derived from Q

  $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + Q(S', A') - Q(S, A)\right]$

  $S \leftarrow S'$

end

CONTROL ALGORITHM: Q-LEARNING

Algorithm : Q-learning to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, arbitrarily (e.g., to zero)
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $A' \leftarrow$ action in S' according to the greedy policy derived from Q
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + Q(S', A') - Q(S, A) \right]$
    $S \leftarrow S'$
end

$$Q(S, A) = Q(S, A) + \alpha \left[R + \max_a Q(S', a) - Q(S, A) \right]$$

CONTROL ALGORITHM: Q-LEARNING

Algorithm : Q-learning to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, arbitrarily (e.g., to zero)
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $A' \leftarrow$ action in S' according to the greedy policy derived from Q
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + Q(S', A') - Q(S, A)\right]$
    $S \leftarrow S'$
end

"Off-policy"

$$Q(S, A) = Q(S, A) + \alpha\left[R + \max_a Q(S', a) - Q(S, A)\right]$$

PLANNING USING A MODEL OF THE WORLD

PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} \underline{p(s', r \mid s, a)} \left[r + v_t(s') \right]$$

PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} \underline{p(s', r \mid s, a)} \left[r + v_t(s') \right]$$

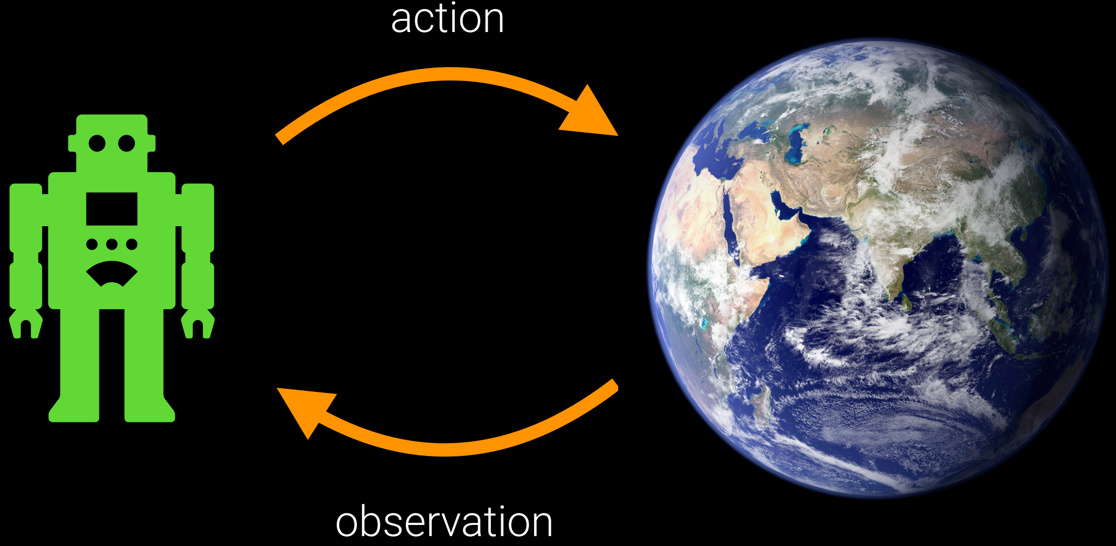
- We can also use the model as a substitute for real-world experience:

PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} \underline{p(s', r \mid s, a)} \left[r + v_t(s')\right]$$

- We can also use the model as a substitute for real-world experience:

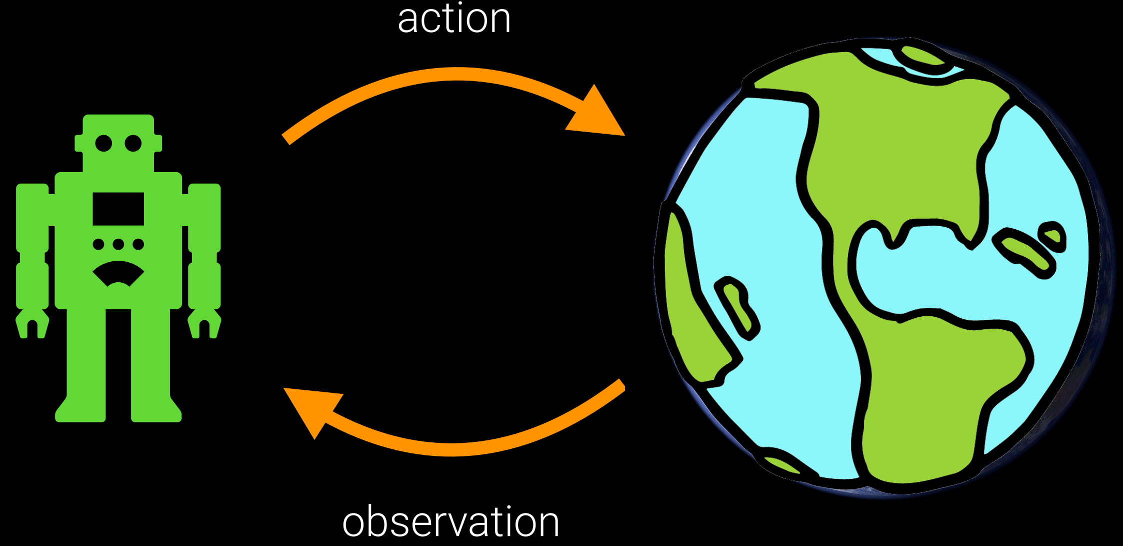


PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} \underline{p(s', r \mid s, a)} \left[r + v_t(s') \right]$$

- We can also use the model as a substitute for real-world experience:



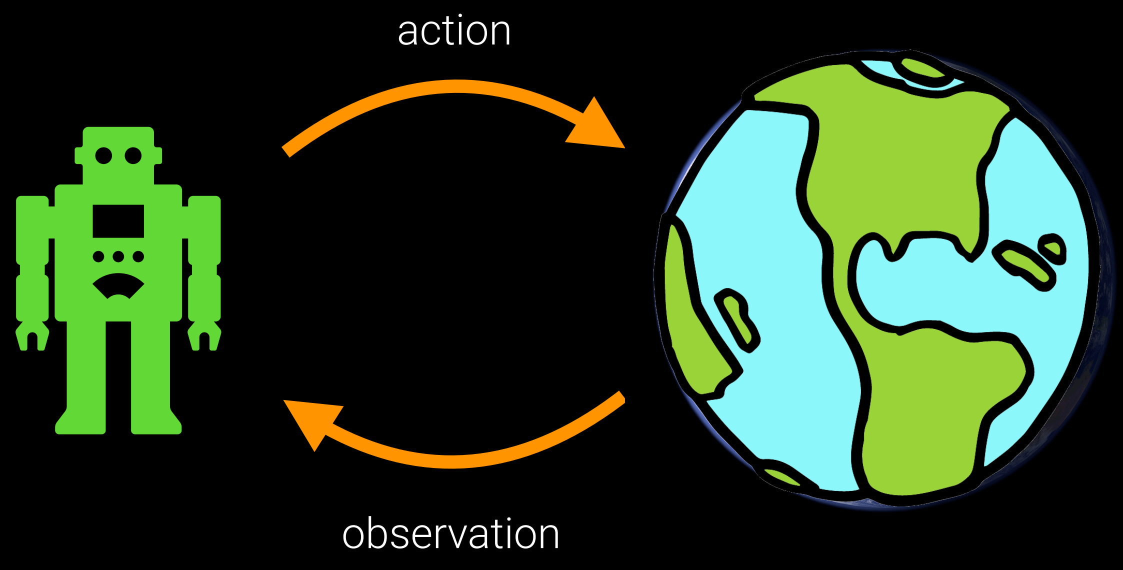
PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} \underline{p(s', r \mid s, a)} \left[r + v_t(s') \right]$$

- We can also use the model as a substitute for real-world experience:

$$v_{t+1}(S) = v_t(S) + \alpha \left[R + v_t(S') - v_t(S) \right]$$



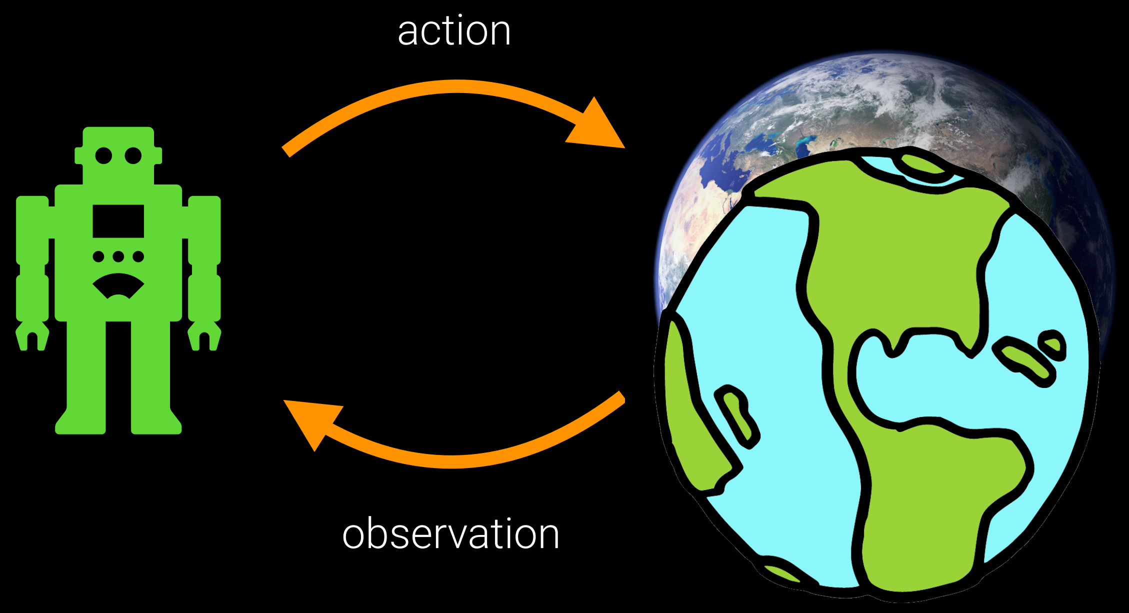
PLANNING USING A MODEL OF THE WORLD

- We studied one way to use a model with DP:

$$v_{t+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} \underline{p(s', r \mid s, a)} \big[r + v_t(s') \big]$$

- We can also use the model as a substitute for real-world experience:

$$v_{t+1}(S) = v_t(S) + \alpha \big[R + v_t(S') - v_t(S) \big]$$



LEARNING A MODEL FROM EXPERIENCE

LEARNING A MODEL FROM EXPERIENCE

- Where do models come from?

LEARNING A MODEL FROM EXPERIENCE

- Where do models come from?
    - They can also be learned from experience.

LEARNING A MODEL FROM EXPERIENCE

- Where do models come from?
  - They can also be learned from experience.

A a 1 B b 1 T
A b 0 C a 3 T
A a 1 B a 0 T
A b 0 C b 0 T

DYNA: INTEGRATING LEARNING AND PLANNING

Learning the model

Planning with
the learned model

DYNA: INTEGRATING LEARNING AND PLANNING

Algorithm : Dyna to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$ and $Model(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    Update model: $Model(S, A) \leftarrow S', R$ — Learning the model
    while *time to plan* **do**
        $S_p \leftarrow$ random previously observed state
        $A_p \leftarrow$ random previously picked action in S
        $R_p, S'_p \leftarrow Model(S_p, A_p)$
        $Q(S_p, A_p) \leftarrow Q(S_p, A_p) + \alpha \left[R_p + \max_a Q(S'_p, a) - Q(S_p, A_p)\right]$
    end
    $S \leftarrow S'$
end

Planning with the learned model

DYNA: INTEGRATING LEARNING AND PLANNING

Algorithm : Dyna to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$ and $Model(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Observe initial state S
for *each time step* **do**
  $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
  Take action A, observe R, S'
  Update model: $Model(S, A) \leftarrow S', R$ — Learning the model
  while *time to plan* **do**
    $S_p \leftarrow$ random previously observed state
    $A_p \leftarrow$ random previously picked action in S
    $R_p, S'_p \leftarrow Model(S_p, A_p)$
    $Q(S_p, A_p) \leftarrow Q(S_p, A_p) + \alpha \left[R_p + \max_a Q(S'_p, a) - Q(S_p, A_p)\right]$
  end
  $S \leftarrow S'$
end

Planning with the learned model

DYNA: INTEGRATING LEARNING AND PLANNING

Algorithm : Dyna to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$ and $Model(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    Update model: $Model(S, A) \leftarrow S', R$ — Learning the model
    while *time to plan* **do**
        $S_p \leftarrow$ random previously observed state
        $A_p \leftarrow$ random previously picked action in S
        $R_p, S'_p \leftarrow Model(S_p, A_p)$
        $Q(S_p, A_p) \leftarrow Q(S_p, A_p) + \alpha \left[R_p + \max_a Q(S'_p, a) - Q(S_p, A_p)\right]$
    end
    $S \leftarrow S'$
end

Planning with the learned model

DYNA: INTEGRATING LEARNING AND PLANNING

Algorithm : Dyna to estimate $Q \approx Q_{\pi^*}$

Parameters: step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$ and $Model(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Observe initial state S
for *each time step* **do**
    $A \leftarrow$ action in S according to policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    Update model: $Model(S, A) \leftarrow S', R$ Learning the model
    while *time to plan* **do**
        $S_p \leftarrow$ random previously observed state
        $A_p \leftarrow$ random previously picked action in S
        $R_p, S'_p \leftarrow Model(S_p, A_p)$
        $Q(S_p, A_p) \leftarrow Q(S_p, A_p) + \alpha \left[R_p + \max_a Q(S'_p, a) - Q(S_p, A_p)\right]$
    end
    $S \leftarrow S'$
end

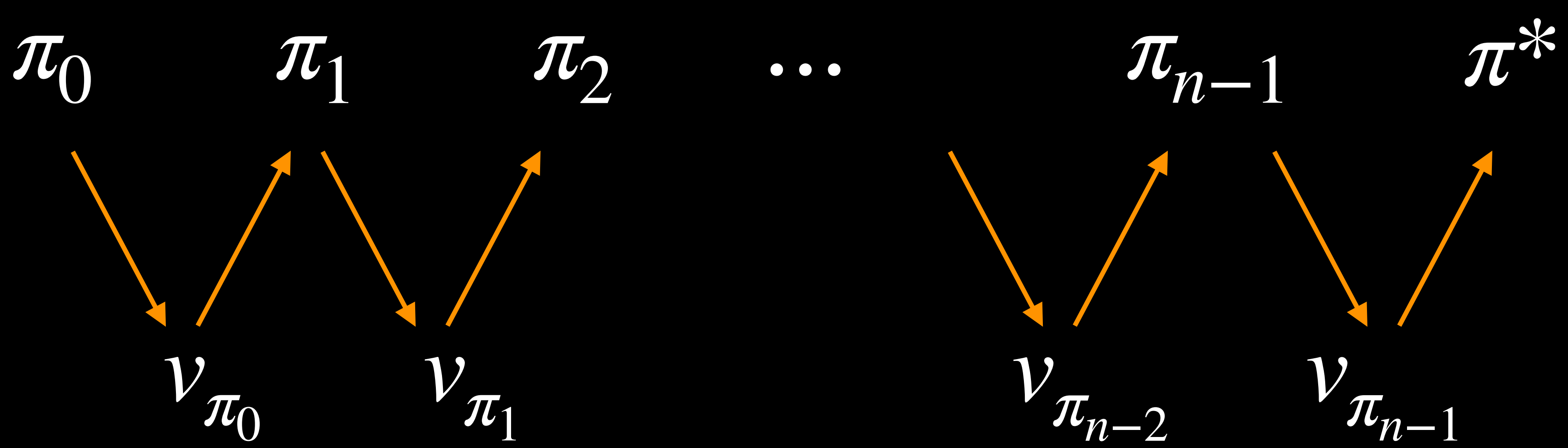
Planning with the learned model

OUTLINE

- Dynamic Programming (DP)
- Temporal-Difference (TD) Learning
- Model-based RL
- Policy Optimization

- So far we have used value estimates to improve the policy.

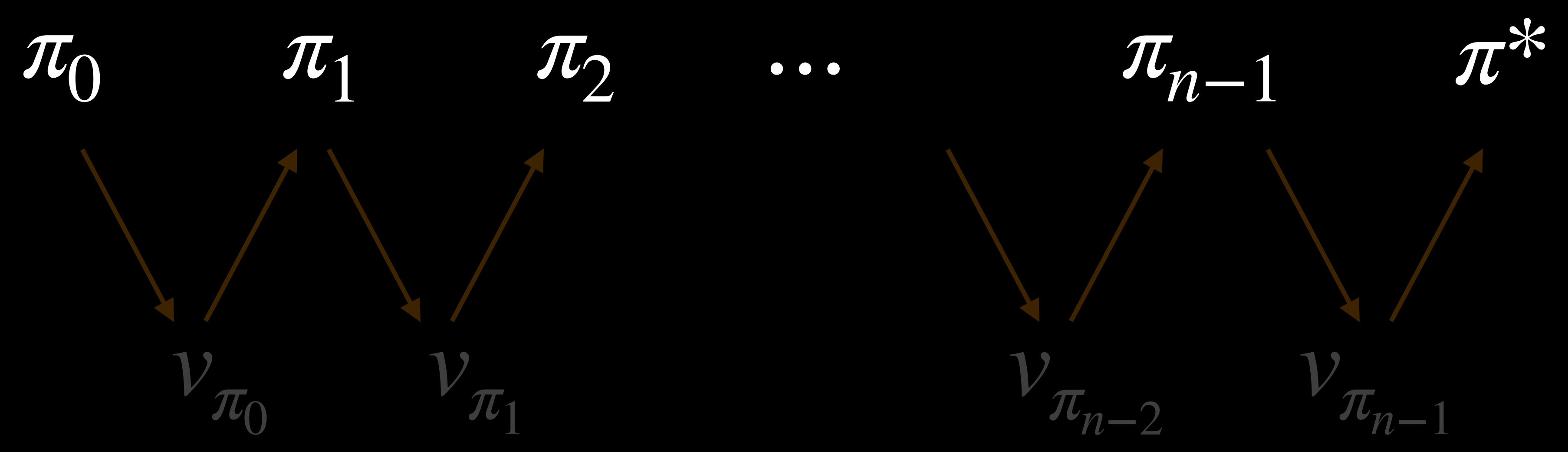
- So far we have used value estimates to improve the policy.



- So far we have used value estimates to improve the policy.
- Can we directly improve the policy?

$$
\begin{tikzcd}
\pi_0 \arrow[dr] & & \pi_1 \arrow[dr] & & \pi_2 & \cdots \arrow[dr] & & \pi_{n-1} \arrow[dr] & & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} \arrow[ur] & & & v_{\pi_{n-2}} \arrow[ur] & & v_{\pi_{n-1}} \arrow[ur] &
\end{tikzcd}
$$

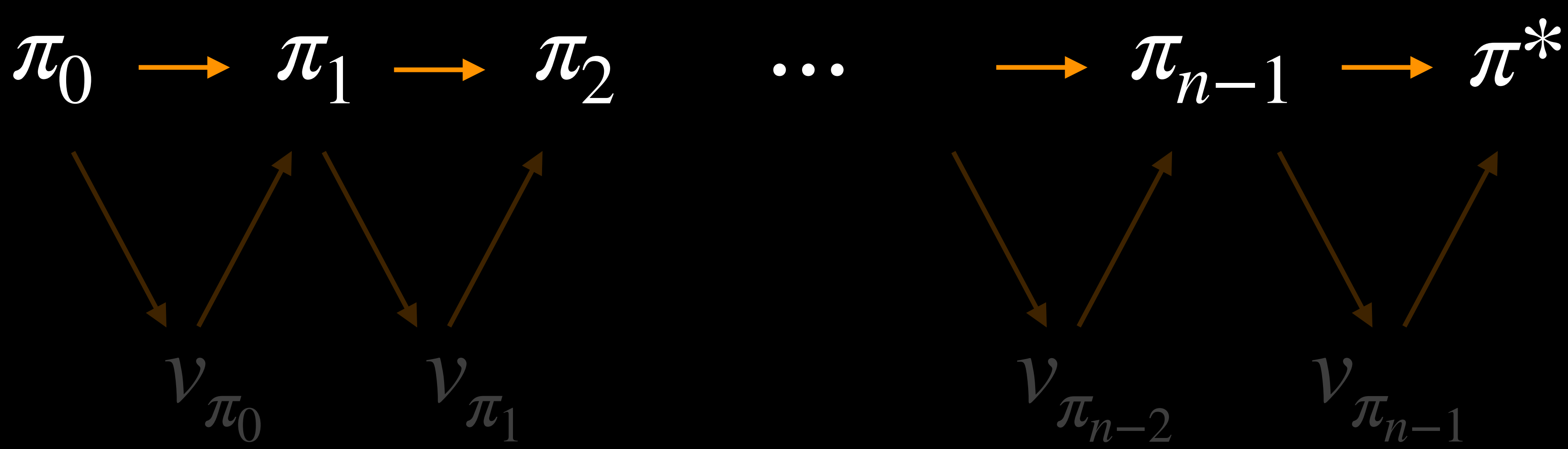
- So far we have used value estimates to improve the policy.
- Can we directly improve the policy?



- So far we have used value estimates to improve the policy.
- Can we directly improve the policy?

$$
\begin{tikzcd}
\pi_0 \arrow[rr] \arrow[dr] & & \pi_1 \arrow[rr] \arrow[dr] & & \pi_2 & \cdots \arrow[rr] \arrow[dr] & & \pi_{n-1} \arrow[rr] \arrow[dr] & & \pi^* \\
& v_{\pi_0} \arrow[ur] & & v_{\pi_1} \arrow[ur] & & & v_{\pi_{n-2}} \arrow[ur] & & v_{\pi_{n-1}} \arrow[ur] &
\end{tikzcd}
$$

- So far we have used value estimates to improve the policy.
- Can we directly improve the policy?



Yes!

POLICY PARAMETRIZATION

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

- In policy-based methods, we need not consult the value function to select actions.

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

- In policy-based methods, we need not consult the value function to select actions.
  - Instead, we use action preferences that are learned.

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

- In policy-based methods, we need not consult the value function to select actions.
  - Instead, we use action preferences that are learned.

$$h(s, a),\ \forall s, a$$

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

- In policy-based methods, we need not consult the value function to select actions.
    - Instead, we use action preferences that are learned.

$$h(s, a), \ \forall s, a \qquad \qquad \pi(a \mid s) = \frac{h(s, a)}{\sum_b h(s, b)}$$

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

- In policy-based methods, we need not consult the value function to select actions.
  - Instead, we use action preferences that are learned.

$$h(s, a), \; \forall s, a \qquad\qquad \pi(a \mid s) = \frac{h(s, a)}{\sum_b h(s, b)} \text{ or } \frac{e^{h(s,a)}}{\sum_b e^{h(s,b)}}$$

POLICY PARAMETRIZATION

- In value-based methods, we select actions based on the learned value function:

$$\pi(s) = \arg\max_a Q(s, a)$$

- In policy-based methods, we need not consult the value function to select actions.
    - Instead, we use action preferences that are learned.

$h(s, a), \ \forall s, a$

$$\pi(a \mid s) = \frac{h(s,a)}{\sum_b h(s,b)} \text{ or } \frac{e^{h(s,a)}}{\sum_b e^{h(s,b)}}$$

"soft-max"

LEARNING ACTION PREFERENCES

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.
- Objective: maximize amount of reward obtained

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.
- Objective: maximize amount of reward obtained
- Method: stochastic gradient ascent

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.
- Objective: maximize amount of reward obtained
- Method: stochastic gradient ascent

$$\pi(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$$

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.
- Objective: maximize amount of reward obtained
- Method: stochastic gradient ascent

$$\pi(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\left(R_t - \bar{R}_t\right)\left(1 - \pi(A_t)\right)$$

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.
- Objective: maximize amount of reward obtained
- Method: stochastic gradient ascent

$$\pi(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\left(R_t - \bar{R}_t\right)\left(1 - \pi(A_t)\right)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\left(R_t - \bar{R}_t\right)\left(0 - \pi(a)\right) \quad \forall a \neq A_t$$

LEARNING ACTION PREFERENCES

- Consider a simple case: there is just one state.
- Objective: maximize amount of reward obtained
- Method: stochastic gradient ascent

$$\pi(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$$

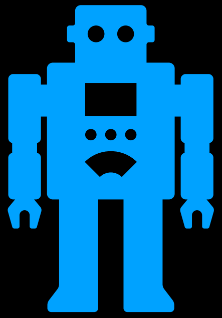
"Gradient-Bandit Algorithm"

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\left(R_t - \bar{R}_t\right)\left(1 - \pi(A_t)\right)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\left(R_t - \bar{R}_t\right)\left(0 - \pi(a)\right) \quad \forall a \neq A_t$$

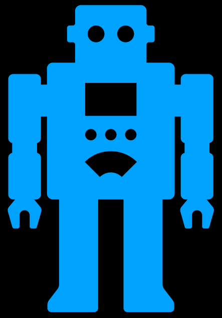
OUTLINE

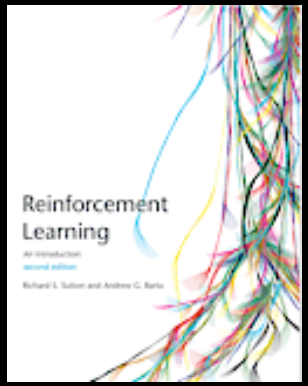
- Dynamic Programming (DP)
- Temporal-Difference (TD) Learning
- Model-based RL
- Policy Optimization



OUTLINE

- Dynamic Programming (DP)
- Temporal-Difference (TD) Learning
- Model-based RL
- Policy Optimization





Further reading:
Sutton & Barto, 2018, *Reinforcement Learning: An Introduction*, 2nd Edition
http://incompleteideas.net/book/the-book.html

THANK YOU

Questions?

abhisheknaik96.github.io
abhishek.naik@ualberta.ca
anaik96