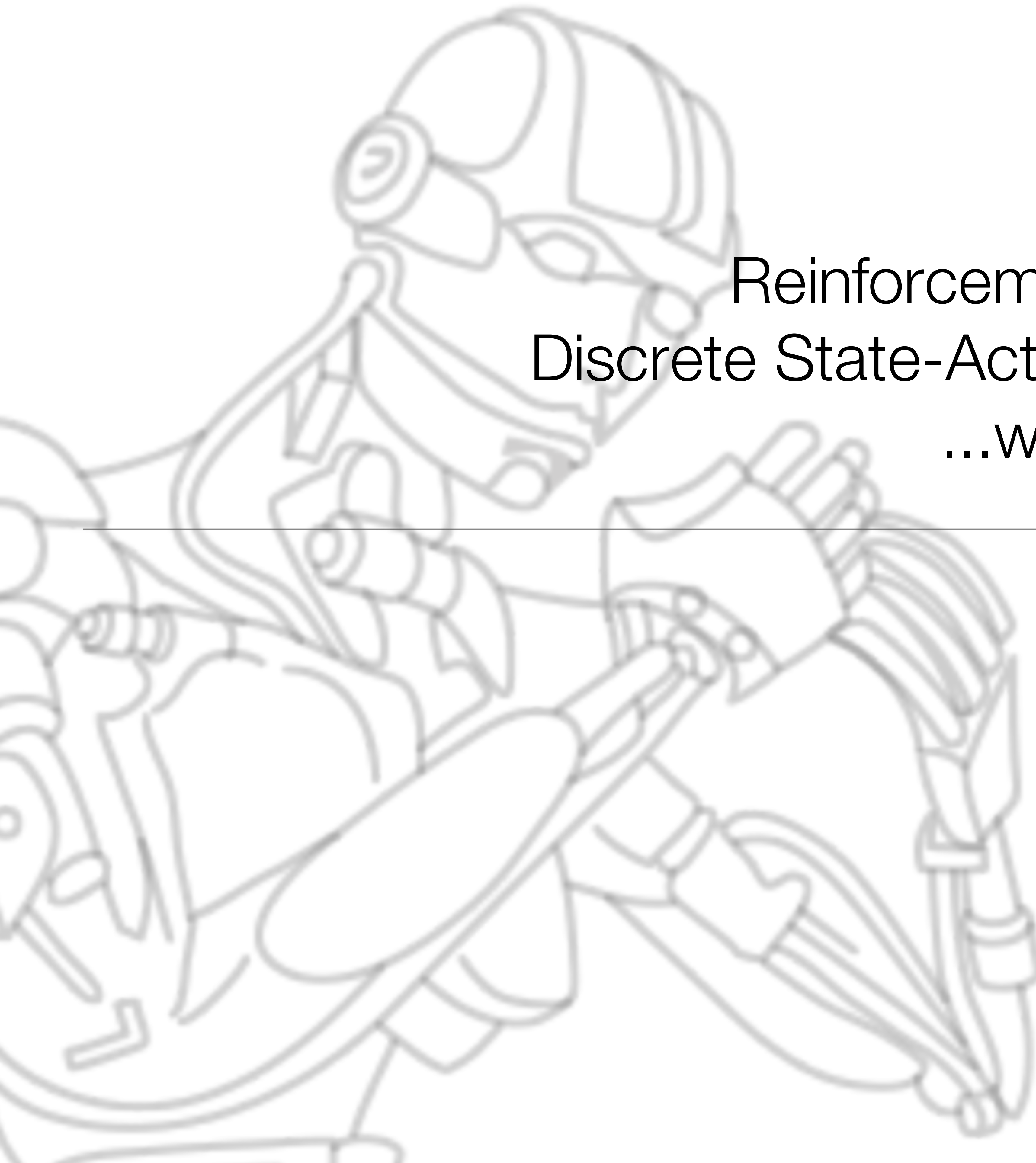


Reinforcement Learning Part I: Discrete State-Action Optimal Control ...with Learned Models

Jan Peters
Gerhard Neumann
Guilherme Maeda

Motivation for optimal decision making in robotics

Typically, **imitation is not enough**

- Imperfect demonstrations
- Correspondance problem
- We can not demonstrate everything

Hence, we need **self-improvment!**

- The robot explores by trial and error
- We give evaluative feedback → reward

Exploration

Reward

Today, we are going to look at the problem of how to take optimal decision that maximize the reward

Note:
reward = - cost
Max(reward) = Min(cost)

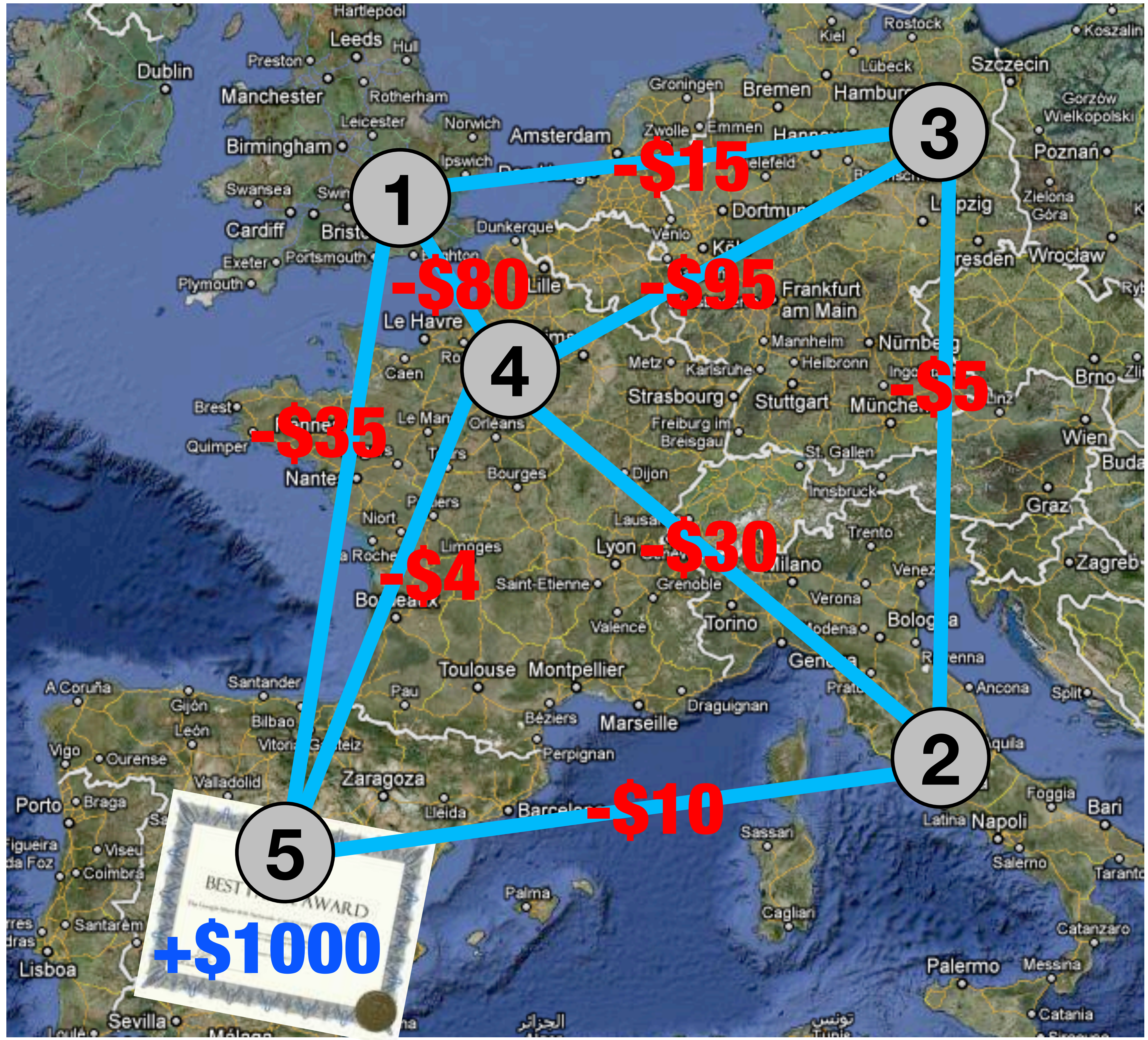
Outline of the Lecture

1. **Introduction**
   - Example of Discrete State-Action Control
   - Formalization of Optimal Control as Markov Decision Process
2. **Finite-Horizon Optimal Control**
   - Value Iteration with a For-Loop
3. **Infinite Horizon Value Iteration**
   - Value Iteration with a Repeat-Until-Loop
4. **Infinite Horizon Policy Iteration**
   - Policy Evaluation: Generate the value function for a fixed policy
   - Policy Improvement: Compute a better policy

Illustration of basic idea…

You have won a Best-Paper Award in Madrid!

What is the Optimal Policy to Collect it?



Dynamic Programming

"An optimal sequence of controls in a multistage optimization problem has the property that ***whatever the initial stage, state and controls are****, the* ***remaining controls*** *must constitute* ***an optimal sequence of decisions for the remaining problem*** *with stage and state resulting from previous controls considered as initial conditions."*

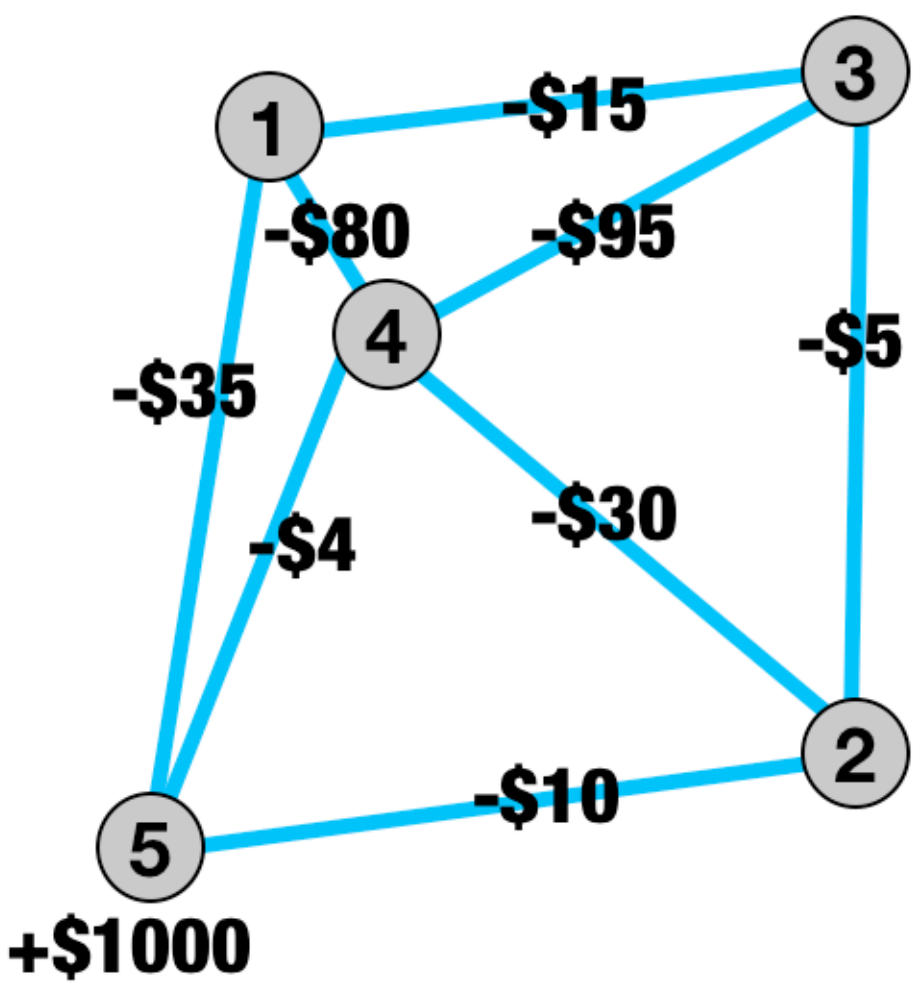
Richard Bellman, Dynamic Programming, 1957

Dynamic Programming

Idea 1: If the optimal solution is broken in small parts, each part must be optimal

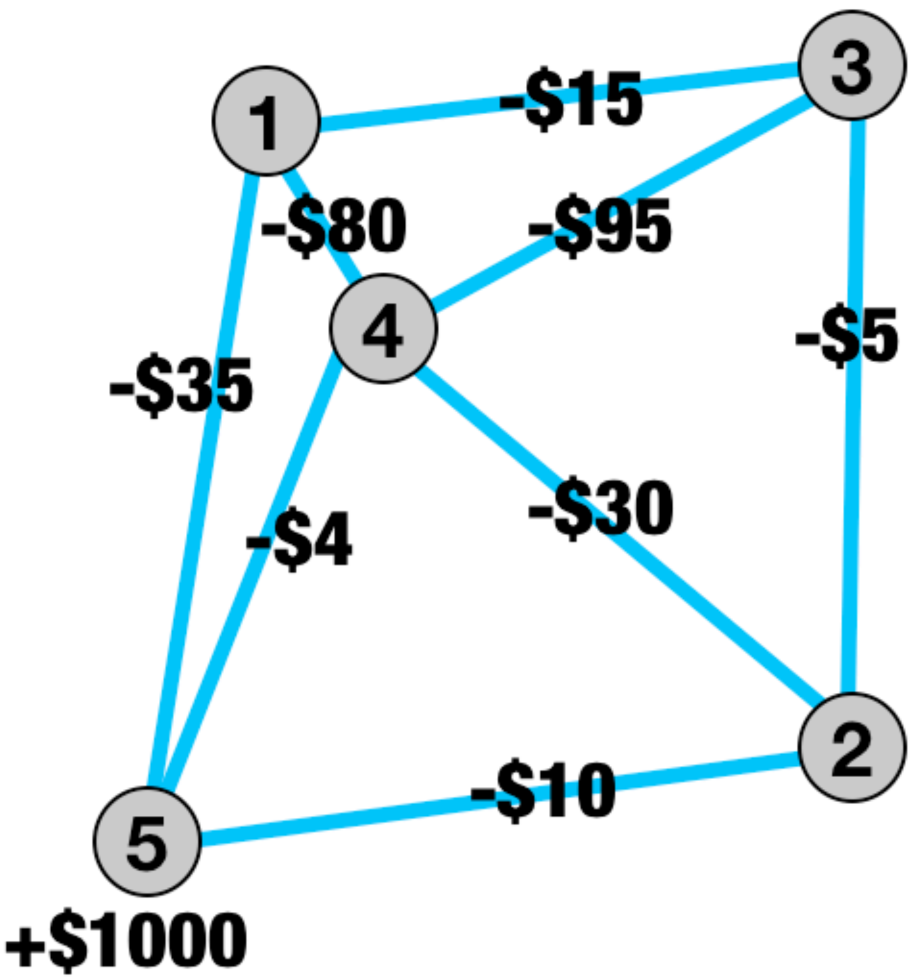
Idea 2: reuse solutions that were already computed (we will store them as V or Q)

Let’s Try this Example!



	T-4	T-3	T-2	T-1	T	
					0	1
					0	2
					0	3
					0	4
					1000	5

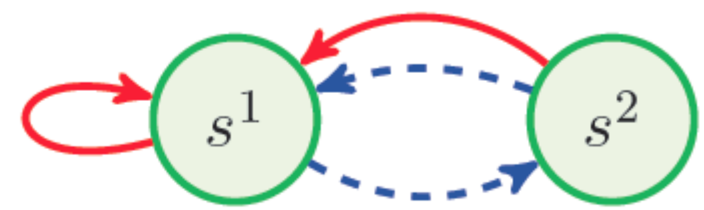
Let’s Try this Example!



	T-4	T-3	T-2	T-1	T	
					0	1
					0	2
					0	3
					0	4
					1000	5

Markov Decision Problems (MDP)

A (non-stationary) **MDP** is defined by:



- its state space $\boldsymbol{s} \in \mathcal{S}$
- its action space $\boldsymbol{a} \in \mathcal{A}$
- its transition dynamics $\mathcal{P}_t(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
- its reward function $r_t(\boldsymbol{s}, \boldsymbol{a})$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$

Markov property:

$$\mathcal{P}_t(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \dots) = \mathcal{P}_t(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$$

- Transition dynamics depends on only on the current time step

Outline of the Lecture

1. **Introduction**
   - Example of Discrete State-Action Control
   - Formalization of Optimal Control as Markov Decision Process
2. **Finite-Horizon Optimal Control**
   - Value Iteration with a For-Loop
3. **Infinite Horizon Value Iteration**
   - Value Iteration with a Repeat-Until-Loop
4. **Infinite Horizon Policy Iteration**
   - Policy Evaluation: Generate the value function for a fixed policy
   - Policy Improvement: Compute a better policy

Finite Horizon Objective

The goal of the agent is to find a policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ that maximizes its expected return $J_{\boldsymbol{\pi}}$ **for a finite time horizon**

Finite Horizon T: Accumulated expected reward for T steps

$$J_{\boldsymbol{\pi}} = \mathbb{E}_{\mu_0, \mathcal{P}, \pi} \left[\sum_{t=1}^{T-1} r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) + r_T(\boldsymbol{s}_T) \right]$$

$r_T(\boldsymbol{s}_T)$ … final reward

Algorithmic Description of Value Iteration

Init: $V_T^*(s) \leftarrow r_T(s),\, t = T$

Repeat $t = t - 1$

Compute Q-Function for time step t (for each state action pair)

$$Q_t^*(s,a) = r_t(s,a) + \sum_{s'} P_t(s'|s,a) V_{t+1}^*(s')$$

Compute V-Function for time step t (for each state)

$$V_t^*(s) = \max_a Q_t^*(s,a)$$

Until t = 1

Return: Optimal policy for each time step

$$\pi_t^*(s) = \text{argmax}_a Q_t^*(s,a)$$

[check animation]

Value Iteration for Finite Horizon

So how does **dynamic programming** work now?

- Start with last layer… (no transition)

$$V_T^*(\boldsymbol{s}) = r_T(\boldsymbol{s})$$

- Iterate backwards in time

$$V_t^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} \left(r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) + \mathbb{E}_{\mathcal{P}}\left[V_{t+1}^*(\boldsymbol{s}_{t+1}) | \boldsymbol{s}_t, \boldsymbol{a}_t \right] \right)$$

- The optimal value function/policy for time step t is obtained after $T - t + 1$ iterations

$$V_T^*(\boldsymbol{s}_T) \Rightarrow V_{T-1}^*(\boldsymbol{s}_{T-1}) \Rightarrow \dots \Rightarrow V_1^*(\boldsymbol{s}_1)$$

What does a finite life-time T imply?

In the finite horizon case, the **time index is part of the state**

- ➡ It matters, how many time steps are left
- ➡ We can only visit each state (including time index) once!
- ➡ We have a layered / multi stage decision problem
- ➡ **The** optimal policy is time-dependent

$$\pi_t^*(\boldsymbol{a}|\boldsymbol{s}) = \pi^*(\boldsymbol{a}|\boldsymbol{s}, t)$$

- ➡ The reward function and the transition model can be time-dependent, i.e.,

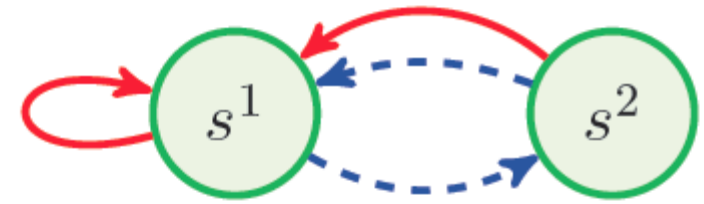
$$r_t(\boldsymbol{s}, \boldsymbol{a}) \text{ and } \mathcal{P}_t(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$$

Outline of the Lecture

1. **Introduction**
   - Example of Discrete State-Action Control
   - Formalization of Optimal Control as Markov Decision Process
2. **Finite-Horizon Optimal Control**
   - Value Iteration with a For-Loop
3. **Infinite Horizon Value Iteration**
   - Value Iteration with a Repeat-Until-Loop
4. **Infinite Horizon Policy Iteration**
   - Policy Evaluation: Generate the value function for a fixed policy
   - Policy Improvement: Compute a better policy

Markov Decision Problems (MDP)

A stationary **MDP** is defined by:



- its state space $\boldsymbol{s} \in \mathcal{S}$
- its action space $\boldsymbol{a} \in \mathcal{A}$
- its transition dynamics $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
- its reward function $r(\boldsymbol{s}, \boldsymbol{a})$ $\quad r_t(\boldsymbol{s}, \boldsymbol{a})$ and $\mathcal{P}_t(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$

$r_t(\boldsymbol{s}, \boldsymbol{a})$ and $\mathcal{P}_t(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$

Markov property:

$$\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \ldots) = \mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$$

- Transition dynamics depends on only on the current time step

So what's special for an infinite horizon, i.e., when you live forever?

In the infinite horizon case, the **time index is not part of the state**

- Your robot lives for all it does, it does not matter, how many time steps are left
- optimal policy is time-independent $\pi_t^*(\boldsymbol{a}|\boldsymbol{s}) = \pi^*(\boldsymbol{a}|\boldsymbol{s})$
- The reward function and the transition model can no longer be time-dependent.
- There is a single, stationary value function for all times.
- We have two different approaches to learning:

1. **Value Iteration:** Use value iteration from before, choose a large *T* for which the value function converges.
2. **Policy Iteration:** Learn a value function for the current policy. Then update the policy and learn a new value function. Repeat.

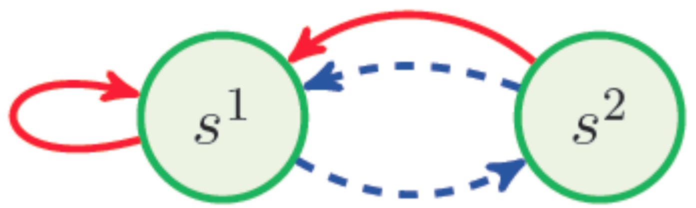
Optimality Objective

The goal of the agent is to find an optimal policy π^* that maximizes its expected long term reward J_π

$$\pi^* = \text{argmax}_\pi J_\pi, \quad J_{\boldsymbol{\pi}} = \mathbb{E}_{\mu_0,\mathcal{P},\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$$

- $0 \leq \gamma < 1$ … discount factor
- Discount Factor trades-off long term vs. immediate reward
- Time Horizon: Infinite

Example: Two State Problem



States: s^1, s^2

Actions: red (a^1) and blue (a^2) edges

Transition:

$$\mathcal{P}(s^1|s^1,a^1)=1,\ \mathcal{P}(s^2|s^1,a^1)=0,\ \mathcal{P}(s^1|s^1,a^2)=0,\ \mathcal{P}(s^2|s^1,a^2)=1$$

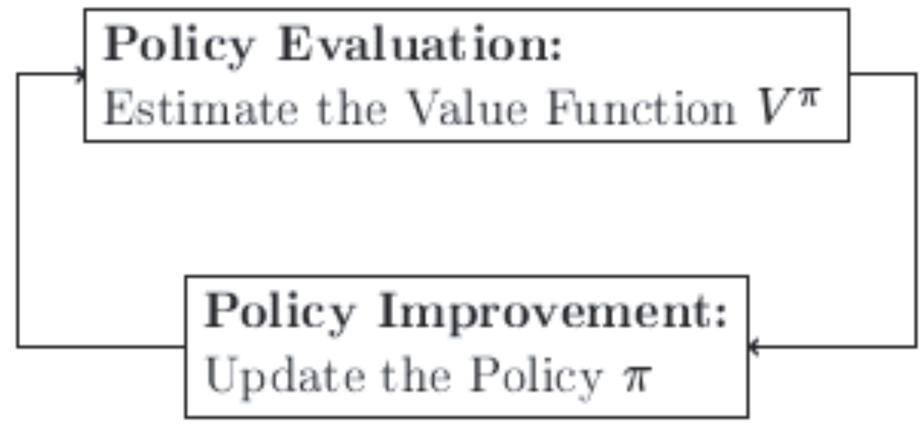
$$\mathcal{P}(s^1|s^2,a^1)=1,\ \mathcal{P}(s^2|s^2,a^1)=0,\ \mathcal{P}(s^1|s^2,a^2)=1,\ \mathcal{P}(s^2|s^2,a^2)=0$$

Rewards: $r(s^1)=1,\ r(s^2)=0$

Policy: What is the optimal infinite horizon policy?

How do we find an optimal policy?

Typically done iteratively:



- Policy Evaluation:

  Estimate quality of states (and actions) with current policy

- Policy Improvement:

  Improve policy by taking actions with the highest quality

Such iterations are called **Policy Iteration**

➔ **will lead to the Bellman Equation ➔ solved by Value Iteration**

Value functions and State-Action Value Functions

Value function $V^{\pi}(\boldsymbol{s})$:

Long-term reward for state $\boldsymbol{s}$ when following policy $\pi(\boldsymbol{a}|\boldsymbol{s})$

$$V^{\pi}(\boldsymbol{s}) = E_{\mathcal{P},\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_0 = \boldsymbol{s}\right]$$

➡ **Quality measure** for state $\boldsymbol{s}$

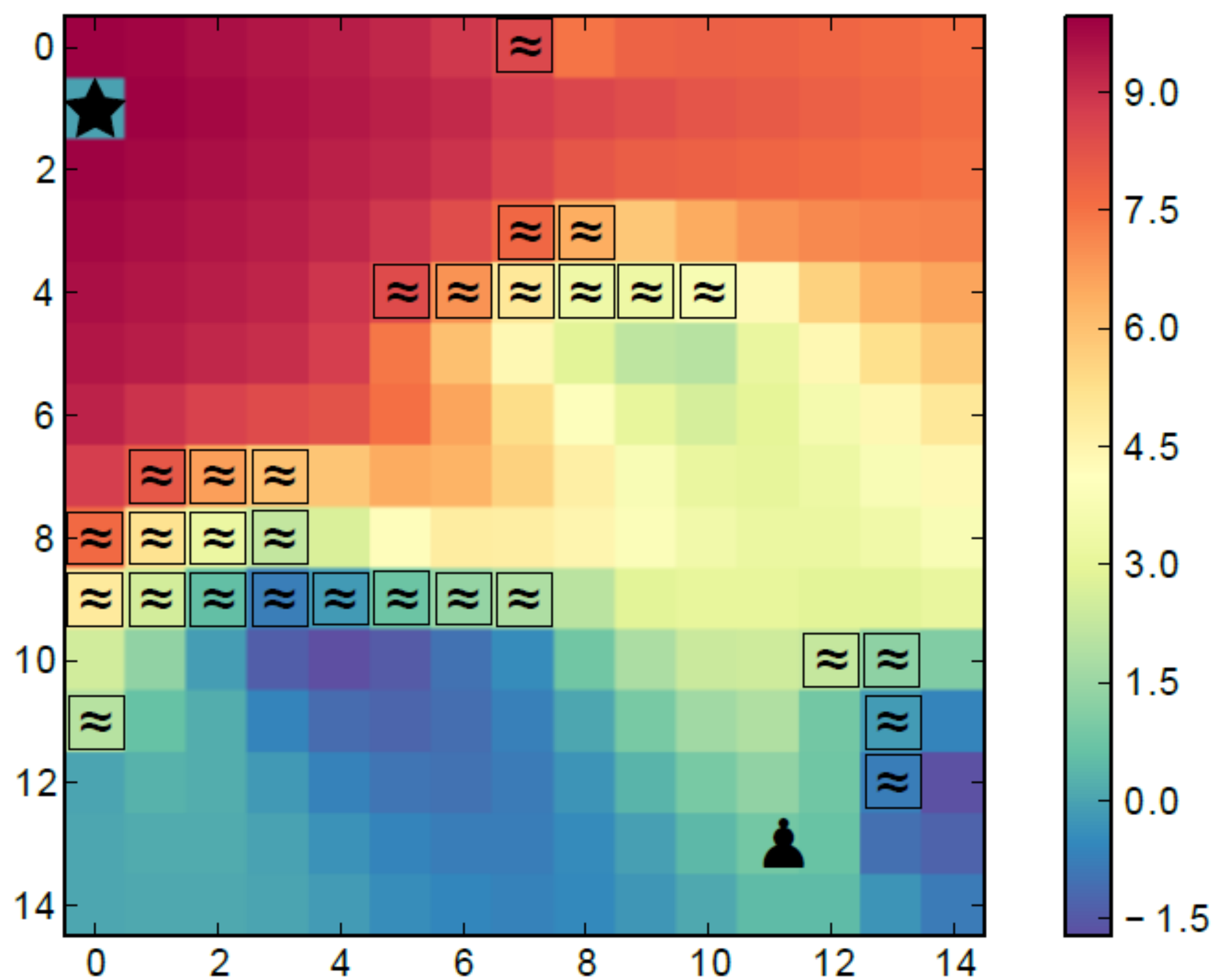
„How good“ is it to be in state s under policy $\pi(a|s)$?

Value functions

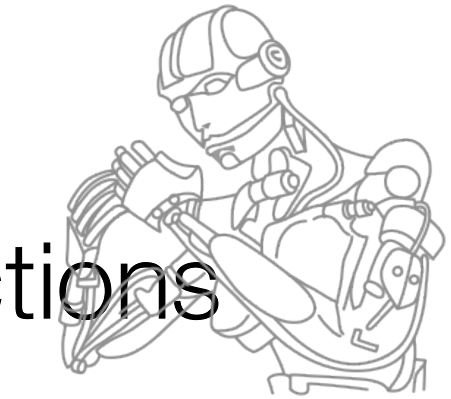
An Illustration…

Policy always goes directly to the star

Going through puddles is punished



Value functions and State-Action Value Functions



Q-function $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$**:**

Long-term reward for taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$ and subsequently following policy $\pi(\boldsymbol{a}|\boldsymbol{s})$

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]$$

➡ **Quality measure** for taking action a in state $\boldsymbol{s}$

„How good“ is it to take action a in state s under policy $\pi(a|s)$?

Value functions and State-Action Value Functions

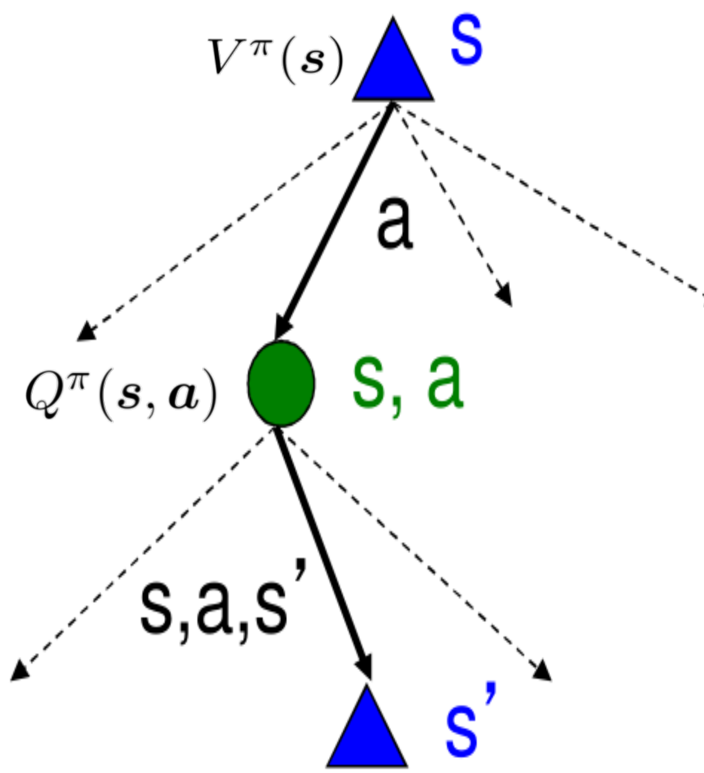
... and can be **easily computed from each other**

Computing V-Function from Q-Function

$$V^\pi(\boldsymbol{s}) = \mathbb{E}_\pi\left[Q^\pi(\boldsymbol{s},\boldsymbol{a})|\boldsymbol{s}\right] = \int \pi(\boldsymbol{a}|\boldsymbol{s})Q^\pi(\boldsymbol{s},\boldsymbol{a})d\boldsymbol{a}$$

Computing Q-Function from V-Function

$$Q^\pi(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \gamma\mathbb{E}_\mathcal{P}\left[V^\pi(\boldsymbol{s}')\middle|\boldsymbol{s},\boldsymbol{a}\right]$$
$$= r(\boldsymbol{s},\boldsymbol{a}) + \gamma\int \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})V^\pi(\boldsymbol{s}')d\boldsymbol{s}'$$



Value functions and State-Action Value Functions

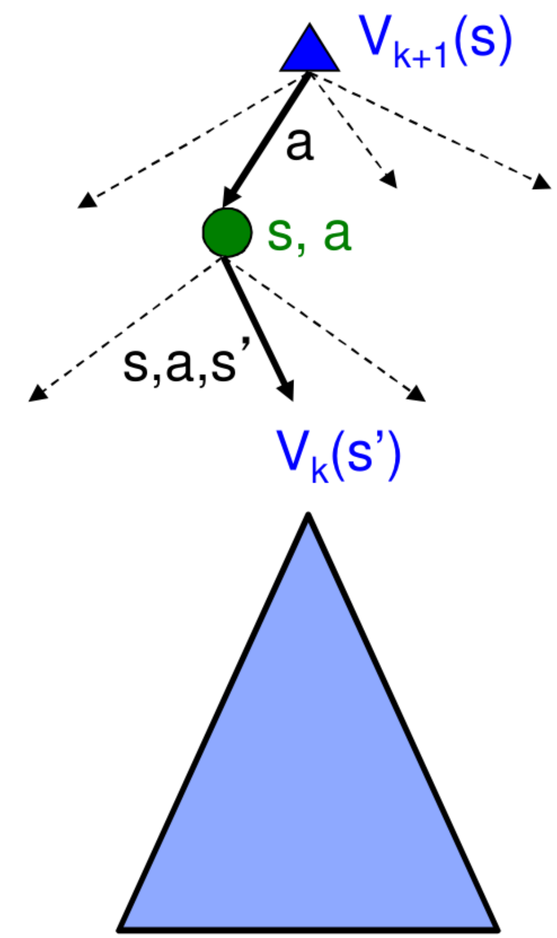
... both functions can also be estimated recursively

$$V^\pi(\boldsymbol{s}) = \mathbb{E}_\pi\left[r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}}\left[V^\pi(\boldsymbol{s}')\right] \middle| \boldsymbol{s}\right]$$

$$= \int \pi(\boldsymbol{a}|\boldsymbol{s})\left(r(\boldsymbol{s},\boldsymbol{a}) + \gamma \int \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) V^\pi(\boldsymbol{s}') d\boldsymbol{s}'\right) d\boldsymbol{a}$$

$$Q^\pi(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P},\pi}\left[Q^\pi(\boldsymbol{s}',\boldsymbol{a}') \middle| \boldsymbol{s},\boldsymbol{a}\right]$$

$$= r(\boldsymbol{s},\boldsymbol{a}) + \gamma \int \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) \int \pi(\boldsymbol{a}'|\boldsymbol{s}') Q^\pi(\boldsymbol{s}',\boldsymbol{a}') d\boldsymbol{a}' d\boldsymbol{s}'$$



➡ If I know the value of the next state $\boldsymbol{s}'$, I can compute the value of the current state

Iterating these equations converges to the true V or Q function

Algorithmic Description of Policy Evaluation

Simplification: For discrete states….

Init: $V_0^\pi(s) \leftarrow 0, \forall s \text{ and } k = 0$

Repeat

Compute Q-Function (for each state action pair)

$$Q_{k+1}^\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k^\pi(s')$$

Compute V-Function (for each state)

$$V_{k+1}^\pi(s) = \sum_a \pi(a|s) Q_{k+1}^\pi(s,a)$$

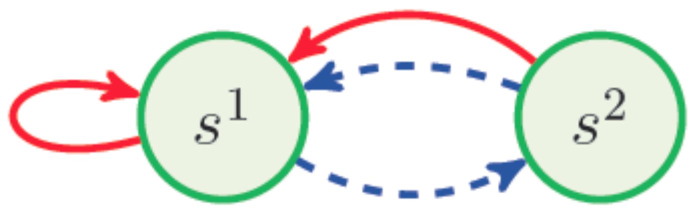
$$k = k + 1$$

until convergence

This algorithm is called Dynamic Programming!

Example: Two State Problem

States: s^1, s^2



Actions: red (a^1) and blue (a^2) edges

Transition:

$$\mathcal{P}(s^1|s^1,a^1)=1,\ \mathcal{P}(s^2|s^1,a^1)=0,\ \mathcal{P}(s^1|s^1,a^2)=0,\ \mathcal{P}(s^2|s^1,a^2)=1$$
$$\mathcal{P}(s^1|s^2,a^1)=1,\ \mathcal{P}(s^2|s^2,a^1)=0,\ \mathcal{P}(s^1|s^2,a^2)=1,\ \mathcal{P}(s^2|s^2,a^2)=0$$

Rewards: $r(s^1)=1,\ r(s^2)=0$

Policy Evaluation: What is the value function of the uniform policy?

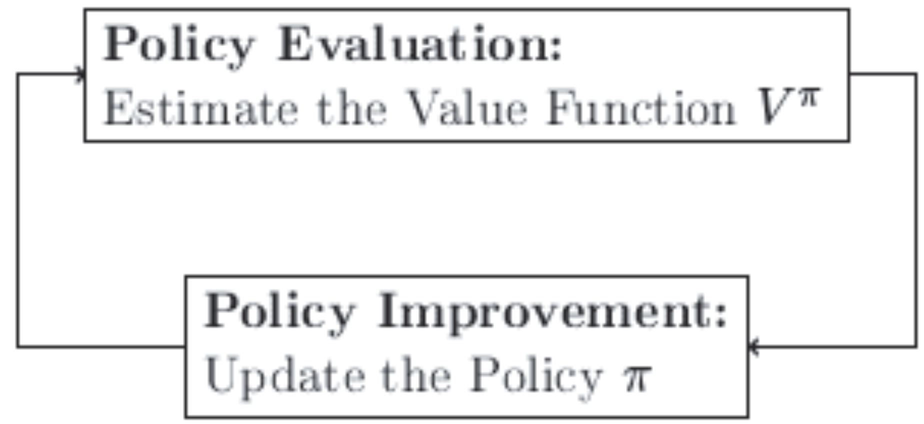
➡ HOMEWORK!

Outline of the Lecture

1. **Introduction**
   - Example of Discrete State-Action Control
   - Formalization of Optimal Control as Markov Decision Process
2. **Finite-Horizon Optimal Control**
   - Value Iteration with a For-Loop
3. **Infinite Horizon Value Iteration**
   - Value Iteration with a Repeat-Until-Loop
4. **Infinite Horizon Policy Iteration**
   - Policy Evaluation: Generate the value function for a fixed policy
   - Policy Improvement: Compute a better policy

How do we find an optimal policy?

Typically done iteratively:



- **Policy Evaluation:**

  Estimate quality of states (and actions) with current policy

- **Policy Improvement:**

  Improve policy by taking actions with the highest quality

  For all states:

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} 1, & \text{if } \boldsymbol{a} = \text{argmax}_{\boldsymbol{a}'} Q^\pi(\boldsymbol{s}, \boldsymbol{a}') \\ 0, & \text{otherwise} \end{cases}$$

Iterating Policy Evaluation and Policy Improvement converges to the optimal policy and is called Policy Iteration

Algorithmic Description of Policy Iteration

Init: $V_0^\pi(s) \leftarrow 0, \pi \leftarrow \text{uniform}$

Repeat

Repeat $k = k + 1$

Compute Q-Function (for each state action pair)

$$Q_{k+1}^\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k^\pi(s')$$

Compute V-Function (for each state)

$$V_{k+1}^\pi(s) = \sum_a \pi(a|s) Q_{k+1}^\pi(s,a)$$

until convergence of V

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} 1, & \text{if } \boldsymbol{a} = \text{argmax}_{\boldsymbol{a}'} Q^\pi(\boldsymbol{s}, \boldsymbol{a}') \\ 0, & \text{otherwise} \end{cases}$$

These are the optimal actions based on values of a suboptimal policy!

until convergence of policy

Value iteration

Can we also **stop policy evaluation before convergence** and perform a policy update?

Yes! We will still converge to the optimal policy !

„Extreme“ case: Stop policy evaluation after 1 iteration

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} \left(r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}} \left[V^*(\boldsymbol{s}') \middle| \boldsymbol{s}, \boldsymbol{a} \right] \right)$$

This equation is called the Bellman Equation

Iterating this equation computes the value function $V^*(\boldsymbol{s})$ of the optimal policy

Value Iteration

Alternatively we can also **iterate Q-functions…**

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}} \left[\max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}', \boldsymbol{a}') \middle| \boldsymbol{s}, \boldsymbol{a} \right]$$

Small side note:

Computing optimal V-Function from optimal Q-Function

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a})$$

Computing optimal Q-Function from optimal V-Function

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}} \left[V^*(\boldsymbol{s}') \middle| \boldsymbol{s}, \boldsymbol{a} \right]$$

Algorithmic Description of Value Iteration

Init: $V_0^*(s) \leftarrow 0$

Repeat $k = k + 1$

Compute Q-Function (for each state action pair)

$$Q_{k+1}^*(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k^*(s')$$

Compute V-Function (for each state)

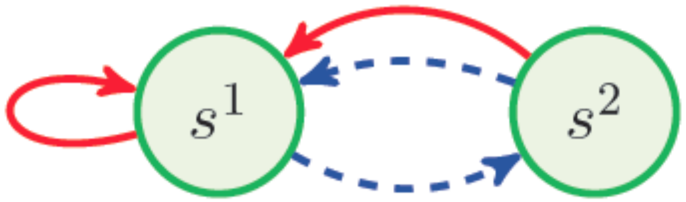
$$V_{k+1}^*(s) = \max_a Q_{k+1}^*(s,a)$$

until convergence of V

[check animation]

Example: Value Iteration

- The Two state example.



HOMEWORK!

Wrap-Up: Dynamic Programming

To compute an optimal policy we can either do…

Policy Iteration:

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}_{\pi}\left[r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}}\left[V^{\pi}(\boldsymbol{s}')\right] \Big| \boldsymbol{s}\right]$$

Policy Evaluation:

Policy Improvement:

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} 1, & \text{if } \boldsymbol{a} = \text{argmax}_{\boldsymbol{a}'} Q^{\pi}(\boldsymbol{s},\boldsymbol{a}') \\ 0, & \text{otherwise} \end{cases}$$

Value Iteration:

Iterate: $$V^{*}(\boldsymbol{s}) = \max_{\boldsymbol{a}}\left(r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}}\left[V^{*}(\boldsymbol{s}') \Big| \boldsymbol{s},\boldsymbol{a}\right]\right)$$

Get optimal policy after convergence:

$$\pi^{*}(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} 1, & \text{if } \boldsymbol{a} = \text{argmax}_{\boldsymbol{a}'} Q^{*}(\boldsymbol{s},\boldsymbol{a}') \\ 0, & \text{otherwise} \end{cases}$$

Outline of the Lecture

1. **Introduction**
   - Example of Discrete State-Action Control
   - Formalization of Optimal Control as Markov Decision Process
2. **Finite-Horizon Optimal Control**
3. **Infinite Horizon Value-Functions**
   - Policy Evaluation for a fixed policy
4. **Computing an Optimal Policy for Any Value Function**
   - Policy Improvement
   - Value iteration

Wrap-Up: Dynamic Programming

We now know how to compute optimal policies for both objectives (finite and infinite horizon)

Cool, thats all we need. Lets go home…

Wait, there is a catch!

Unfortunately, we can only do this in 2 cases

- Discrete Systems

  Easy: integrals turn into sums

  …but the world is not discrete!

- Linear Systems, Quadratic Reward, Gaussian Noise (LQR) (next lecture)

  … but the world is not linear!

Wrap-Up: Dynamic Programming

In all other cases, we have to use approximations!

Why?

1. **Representation of the V-function:**

   How to represent V in continuous state spaces?

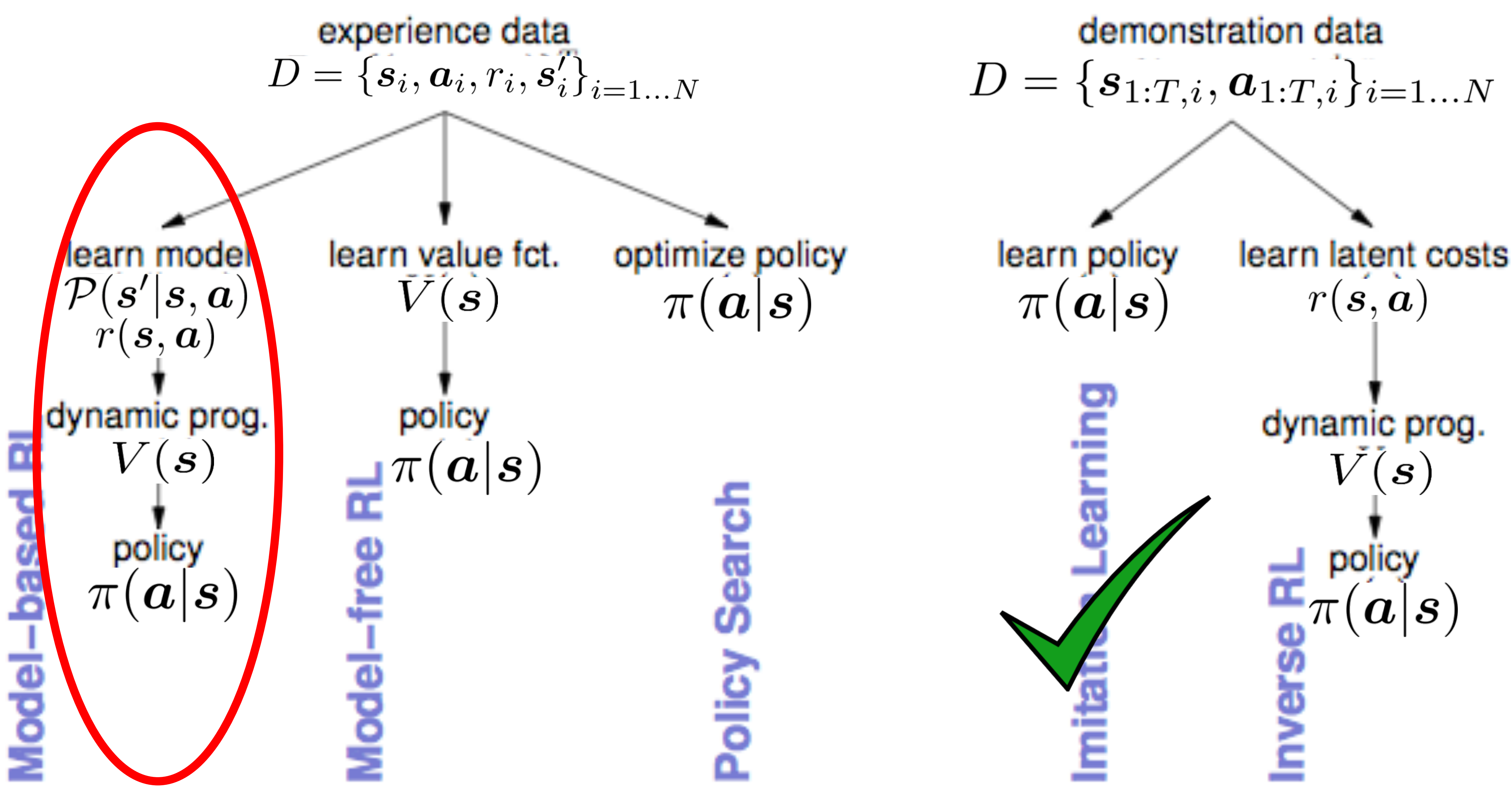
2. **We need to solve:**

   $\max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a})$ **:** difficult in continuous action spaces

   $\mathbb{E}_{\mathcal{P}}\left[V^*(\boldsymbol{s}') \middle| \boldsymbol{s}, \boldsymbol{a}\right]$ **:** difficult for arbitrary functions V and models $\mathcal{P}$

We will hear about that in the next lectures….!

The Bigger Picture: How to learn policies



1. Next Lecture **2.** **3.** **4.**

Discrete State-Action Optimal Control: Summary

What you should know…

- What is a MDP, a value function and a state-action value function…
- What is policy evaluation, policy improvement, policy iteration and value iteration
- The Bellman equation
- Differences of finite and infinite horizon objectives
- Why is it difficult?