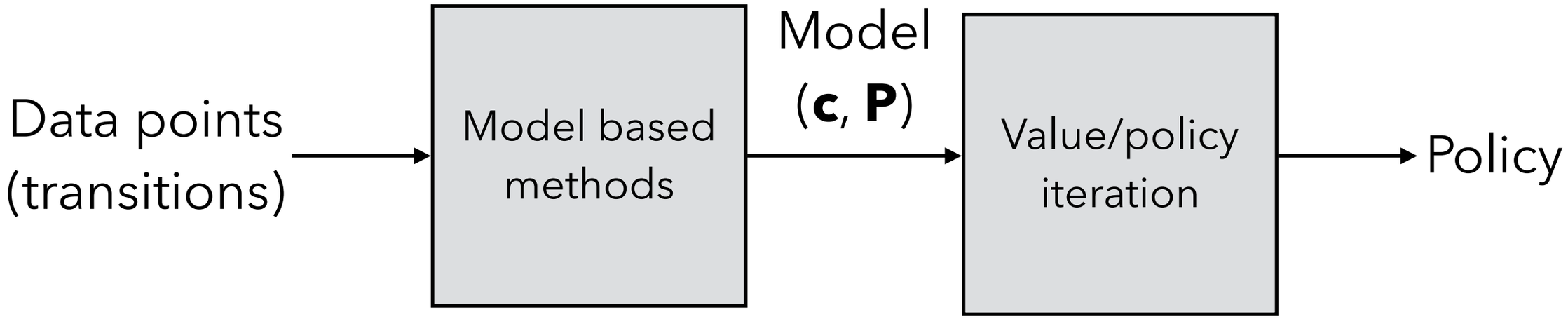
TÉCNICO LISBOA

Planning, Learning and Decision Making

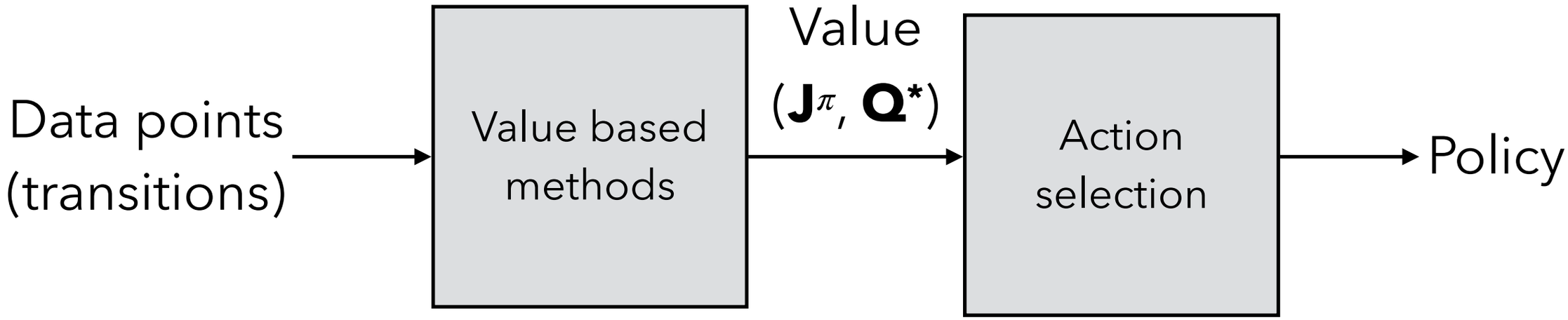
Lecture 20. Reinforcement learning: Policy-gradient

Model based RL



Value based RL

- Value-based methods:



Q-learning

- Given a sample (x_t, a_t, c_t, x_{t+1}),
- Compute

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t \left[c_t + \gamma \min_{a' \in \mathcal{A}} Q_t(x_{t+1}, a') - Q_t(x_t, a_t) \right]$$

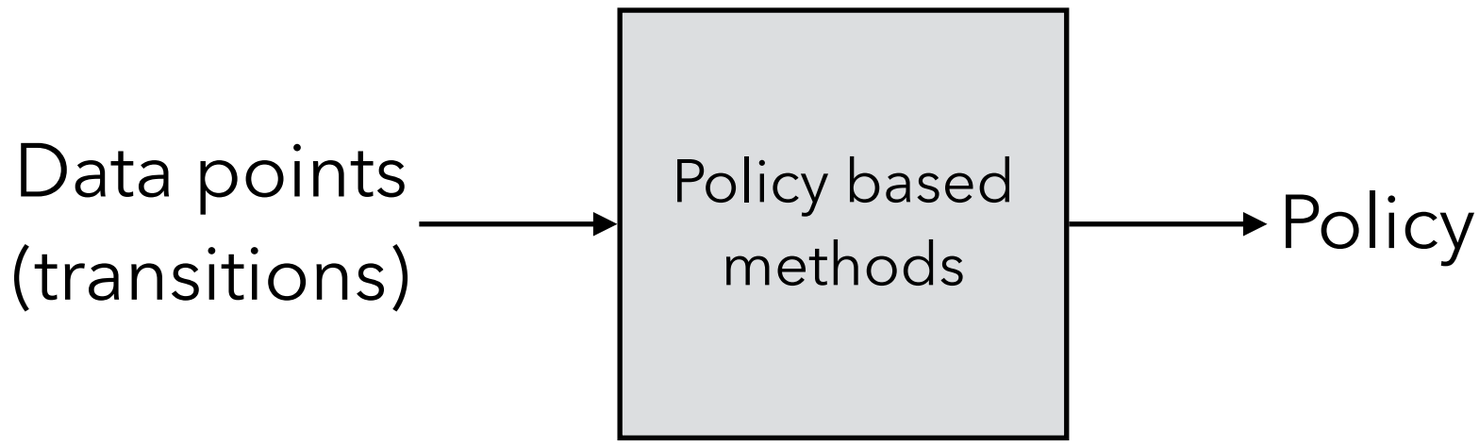
SARSA

- Given a sample $(x_t, a_t, c_t, x_{t+1}, a_{t+1})$,
- Compute

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t \left[c_t + \gamma Q_t(x_{t+1}, a_{t+1}) - Q_t(x_t, a_t)\right]$$

Policy-based RL

- Policy-based methods:



Policy-based methods

Why policy-based RL?

- As seen in previous classes,
  - … value-based methods are not guaranteed to converge with approximations
  - … quality of the result depends on the quality of approximation

Even if approximation is good, policy may be bad

Example

- In some MDP, the Q-values in state x are

a_1	a_2
5	7

- Which is the optimal action?
  - a_1 - minimum cost

Example

- In some MDP, the Q-values in state x are

a_1	a_2
5	6

- Consider the two approximations

a_1	a_2
6	5.5

a_1	a_2
0	1

Which one has smaller error?

Example

- In some MDP, the Q-values in state x are

a_1	a_2
5	6

- Consider the two approximations

a_1	a_2
6	5.5

a_1	a_2
0	1

Which one is better?

Why policy-based RL?

- As seen in previous classes,
  - … value-based methods are not guaranteed to converge with approximations
  - … quality of the result depends on the quality of approximation
  - **… even if the approximation is good, the resulting policy may be bad**

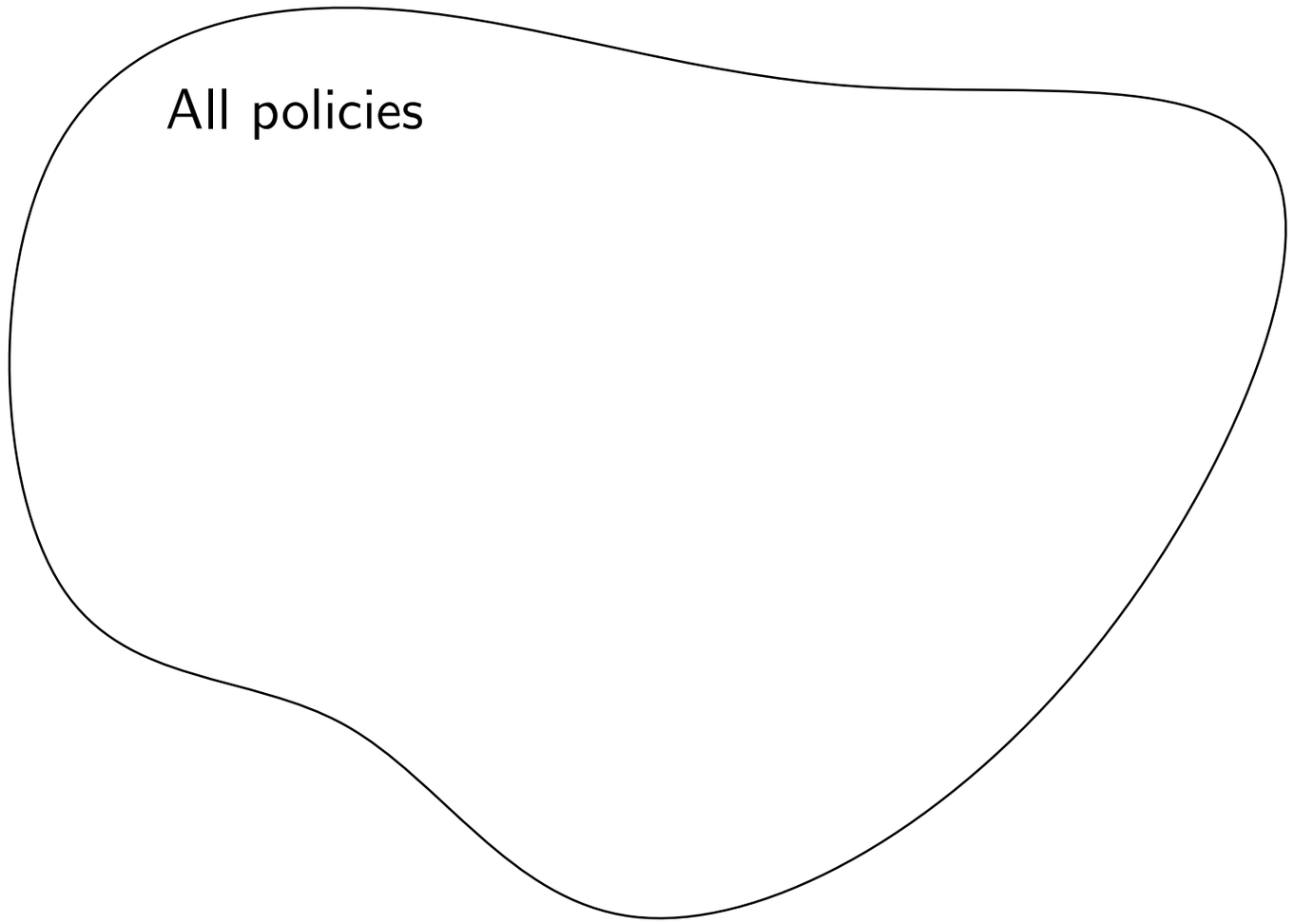
What is policy-based RL?

- Instead of selecting an approximation for the value, we select an approximation for the policy
- Policy-based methods select the best policy within the approximation

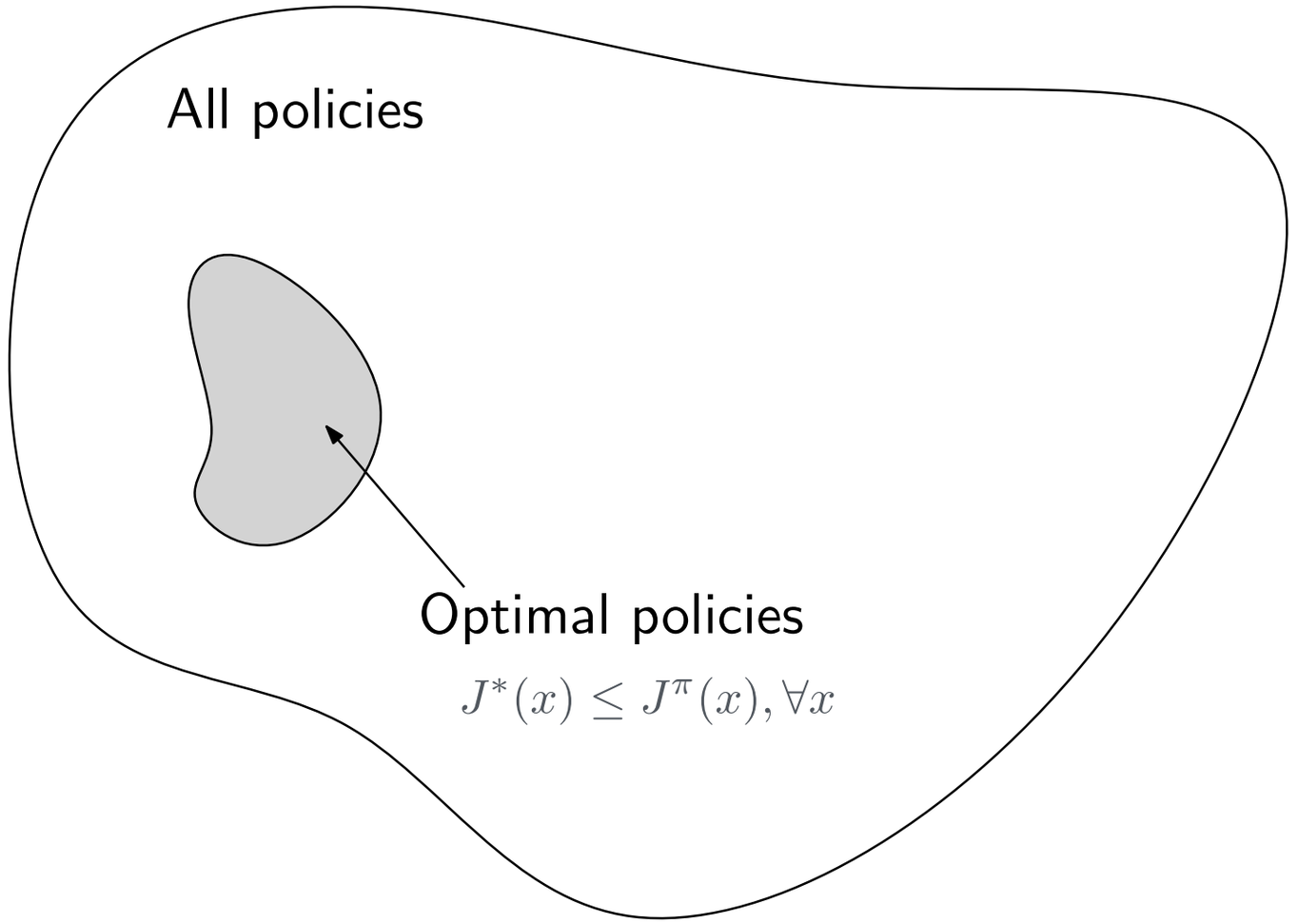
TÉCNICO LISBOA

What is the “best policy”?

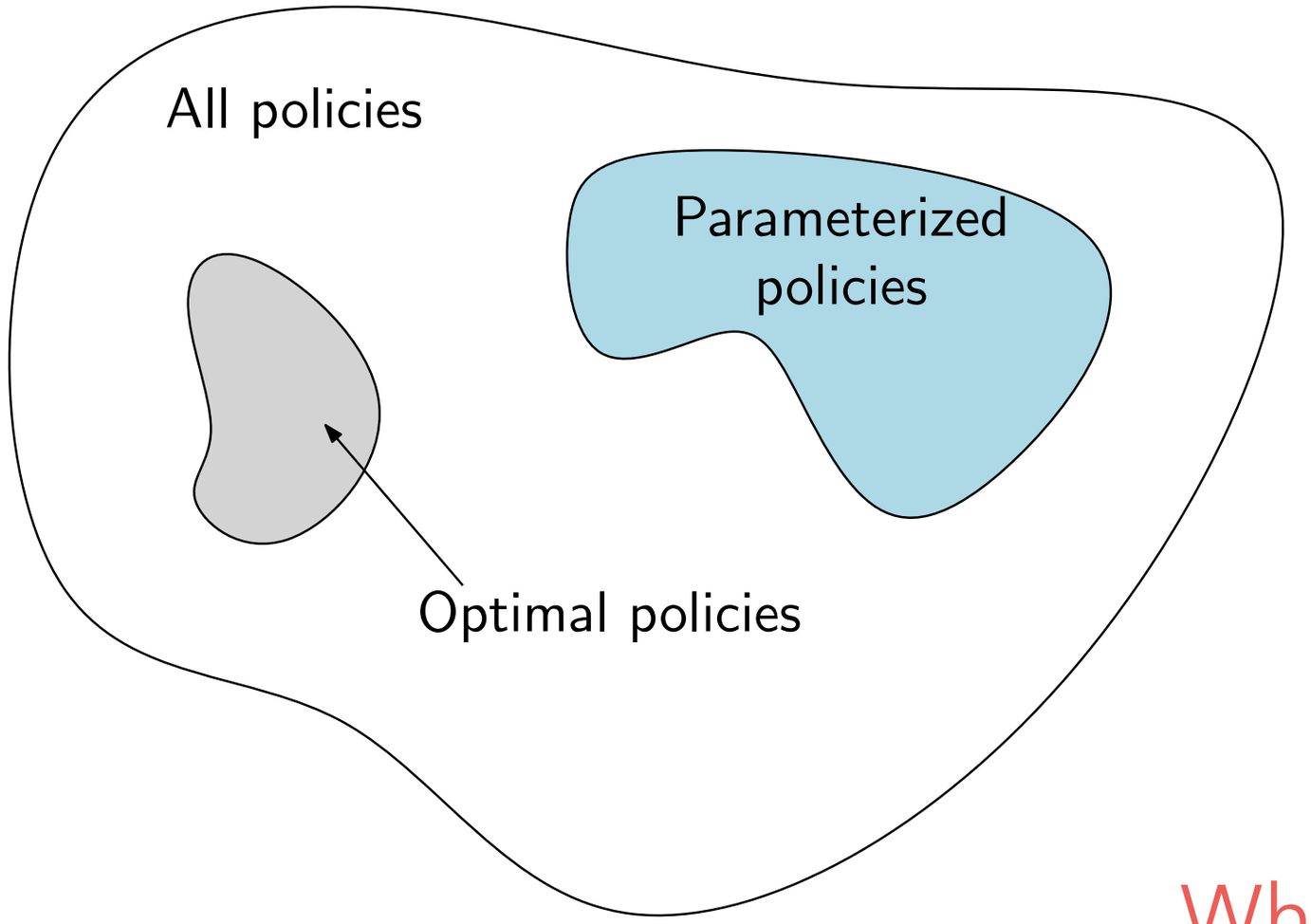
Revisiting optimality



Revisiting optimality



Revisiting optimality



What is “best”?

Example

- Suppose, that, for two policies π_1 and π_2

$$\boldsymbol{J}^{\pi_1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \boldsymbol{J}^{\pi_2} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Which one is “best”?

Initial distribution

- We consider the initial distribution μ_0 and set

$$V(\pi) = \mathbb{E}_{\mu_0}\left[J^\pi(\mathrm{x}_0)\right]$$

$$= \sum_{x \in \mathcal{X}} \mu_0(x) J^\pi(x)$$

↓

Single number

Initial distribution

- We consider the initial distribution μ_0 and set

$$V(\pi) = \mathbb{E}_{\mu_0}\left[J^\pi(\mathrm{x}_0)\right]$$

$$= \sum_{x \in \mathcal{X}} \mu_0(x) J^\pi(x)$$

- Then, π_1 is better than π_2 if $V(\pi_1) < V(\pi_2)$

Policy improvement

- Several approaches:
  - Hill climbing
  - Evolutionary algorithms
  - …

BBO (Black box optimization)

- We focus on **gradient descent:**
  - More efficient
  - Exploits MDP structure better

Policy gradient

- We consider a family of parameterized policies $\{\pi_\theta\}$
- θ is the parameter of the policy
- Example:

$$\pi_{\boldsymbol{\theta}}(a \mid x) = \frac{e^{\boldsymbol{\theta}^\top \boldsymbol{\phi}(x,a)}}{\sum_{a' \in \mathcal{A}} e^{\boldsymbol{\theta}^\top \boldsymbol{\phi}(x,a')}}$$

  … many other possibilities

- V is a function of θ

How to compute the gradient?

Policy gradient

- **Approach 1:** Finite differences

$$\frac{\partial V}{\partial \theta_n} \approx \frac{V(\theta + \varepsilon_n) - V(\theta)}{\varepsilon_n}$$

- Simple to compute
- Works even if policy is not differentiable
- Noisy and slow learning
- Requires evaluation of V

Policy gradient

- **Approach 2:** Analytical
  - Assume that we know $\nabla_\theta \pi$
  - Then,

$$\nabla_\theta V(\theta) = \nabla_\theta \boldsymbol{\mu}_0 \boldsymbol{J}^{\pi_\theta}$$

Does not depend on θ

Policy gradient

- **Approach 2:** Analytical
  - Assume that we know $\nabla_\theta \pi$
  - Then,

$$\nabla_\theta V(\theta) = \boldsymbol{\mu}_0 \nabla_\theta \boldsymbol{J}^{\pi_\theta}$$

Let us consider this term

Policy gradient

- Cost-to-go function verifies

$$J^{\pi_\theta}(x) = \sum_a \pi_\theta(a \mid x) Q^\pi(x,a)$$

$$= \sum_a \pi_\theta(a \mid x) \left[c(x,a) + \gamma \sum_y \mathsf{P}(y \mid x,a) J^{\pi_\theta}(y) \right]$$

Do not depend on θ

Policy gradient

- Cost-to-go function verifies

$$J^{\pi_\theta}(x) = \sum_a \pi_\theta(a \mid x) Q^\pi(x,a)$$

$$= \sum_a \pi_\theta(a \mid x) \left[c(x,a) + \gamma \sum_y \mathsf{P}(y \mid x,a) J^{\pi_\theta}(y) \right]$$

- Then (derivative of the product),

$$\nabla_\theta J^{\pi_\theta}(x) = \sum_a \nabla_\theta \pi_\theta(a \mid x) Q^{\pi_\theta}(x,a) + \gamma \sum_y \mathsf{P}_{\pi_\theta}(y \mid x) \nabla_\theta J^{\pi_\theta}(y)$$

Linear equation

Policy gradient

- We can solve it!

$$\nabla_\theta \boldsymbol{J}^{\pi_\theta} = \nabla_\theta \boldsymbol{\pi}_\theta \boldsymbol{Q}^{\pi_\theta} + \gamma \mathrm{P}_{\pi_\theta} \nabla_\theta \boldsymbol{J}^{\pi_\theta}$$

Statewise dot-product

Policy gradient

- We can solve it!

$$(\boldsymbol{I} - \gamma \mathrm{P}_{\pi_\theta}) \nabla_\theta \boldsymbol{J}^{\pi_\theta} = \nabla_\theta \boldsymbol{\pi}_\theta \boldsymbol{Q}^{\pi_\theta}$$

Policy gradient

- We can solve it!

$$\nabla_\theta \boldsymbol{J}^{\pi_\theta} = (\boldsymbol{I} - \gamma \mathrm{P}_{\pi_\theta})^{-1} \nabla_\theta \boldsymbol{\pi}_\theta \boldsymbol{Q}^{\pi_\theta}$$

- We finally get:

$$\nabla_\theta V(\theta) = \boldsymbol{\mu}_0 (\boldsymbol{I} - \gamma \mathrm{P}_{\pi_\theta})^{-1} \nabla_\theta \boldsymbol{\pi}_\theta \boldsymbol{Q}^{\pi_\theta}$$

μ_θ

Policy gradient theorem

- Continuing the computations, we get the following result.

Theorem: The policy gradient is given by

$$\nabla_\theta V(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a \mid x) Q^{\pi_\theta}(x, a).$$

where μ_θ is a long-term distribution over $\mathcal{X}$.

$$\nabla_\theta \pi_\theta(a \mid x) = \pi_\theta(a \mid x) \frac{\nabla_\theta \pi_\theta(a \mid x)}{\pi_\theta(a \mid x)}$$

Policy gradient theorem

- Continuing the computations, we get the following result.

Theorem: The policy gradient is given by

$$\nabla_\theta V(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a \mid x) Q^{\pi_\theta}(x, a).$$

where μ_θ is a long-term distribution over $\mathcal{X}$.

$$\nabla_\theta \pi_\theta(a \mid x) = \pi_\theta(a \mid x) \nabla_\theta \log \pi_\theta(a \mid x)$$

Policy gradient theorem

- Continuing the computations, we get the following result.

Theorem: The policy gradient is given by

$$\nabla_\theta V(\theta) = \mathbb{E}_{\mu_\theta, \pi_\theta} \left[\nabla_\theta \log \pi_\theta(\mathrm{a} \mid \mathrm{x}) Q^{\pi_\theta}(\mathrm{x}, \mathrm{a}) \right].$$

where μ_θ is a long-term distribution over $\mathcal{X}$.

Must still compute Q^π

REINFORCE Algorithm

- Initialize θ
- Collect a trajectory $\{x_0, a_0, c_0, x_1, a_1, c_1, \ldots, x_T, a_T, c_T\}$ using π_θ
- For each t = 0, …, T
  - Update $\theta \leftarrow \theta - \alpha \nabla \log \pi(a_t \mid x_t) \boxed{\sum_{\tau=t}^{T} \gamma^{\tau+t} c_\tau}$

Estimate of $\gamma^t Q^\pi$

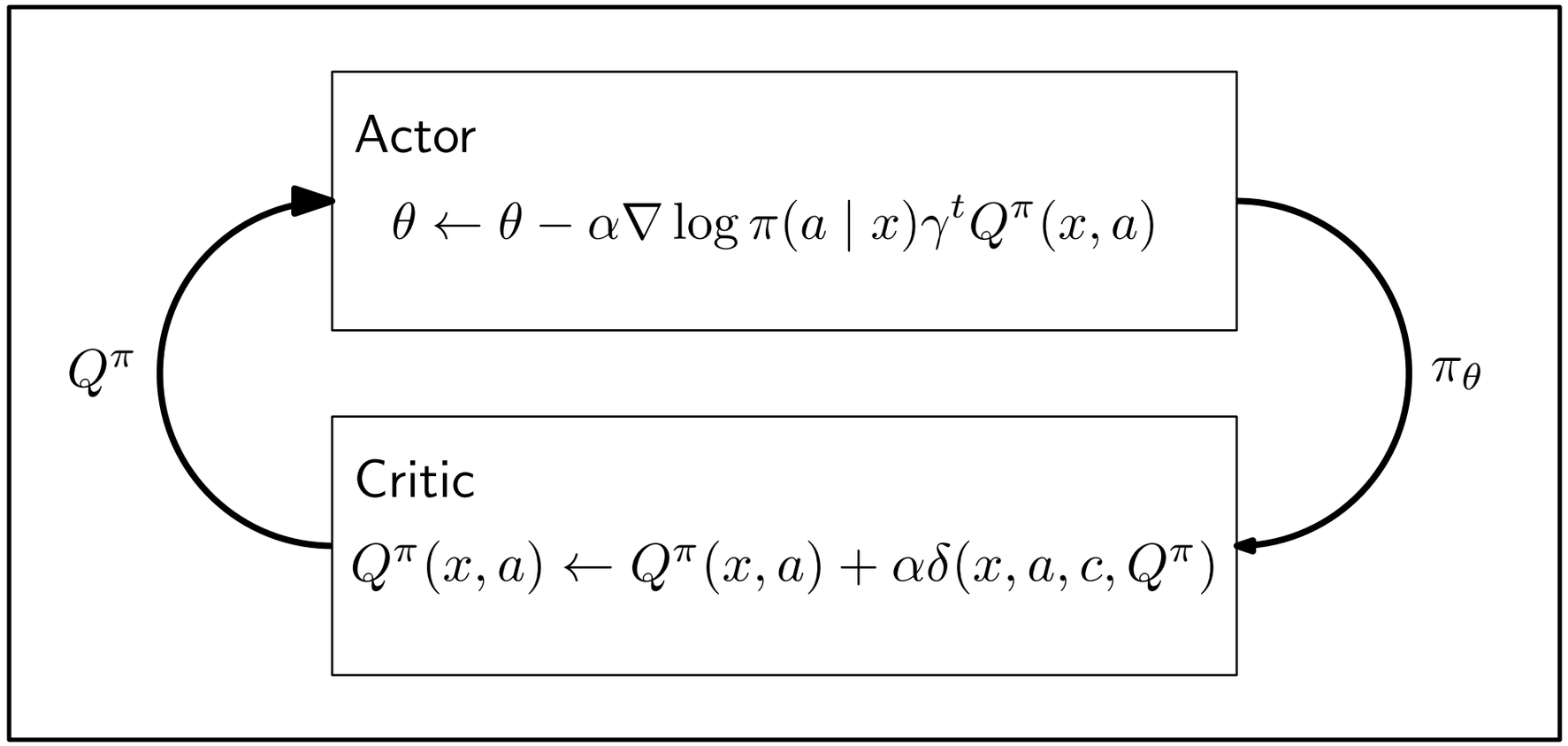
REINFORCE Algorithm

- The REINFORCE algorithm is significantly better than the finite difference approach
- However, gradient estimates have large variance
- Why not use better algorithm to estimate Q^{π}?

Actor-critic architecture

- Split algorithm in two parts:
  - One part maintains/runs the policy (actor)
  - One part computes Q (critic)

Actor-critic architecture



Actor-critic architecture

- However,
  - … we need to estimate Q^{π}
  - … in large domains, we are back to the problems of selecting good approximations for Q^{π}

TÉCNICO LISBOA

… are we?

Policy gradient revisited

- The formula for the gradient gives the answer:

$$\nabla_\theta V(\theta) = \mathbb{E}_{\mu_\theta, \pi_\theta} \left[\nabla_\theta \log \pi_\theta(\mathrm{a} \mid \mathrm{x}) Q^{\pi_\theta}(\mathrm{x}, \mathrm{a}) \right].$$

Don’t really need Q^π

Policy gradient revisited

- The formula for the gradient gives the answer:

$$\nabla_\theta V(\theta) = \mathbb{E}_{\mu_\theta, \pi_\theta} \left[\nabla_\theta \log \pi_\theta(\mathrm{a} \mid \mathrm{x}) Q^{\pi_\theta}(\mathrm{x}, \mathrm{a}) \right].$$

Inner product

Compatible features

- We let

$$\phi(x, a) = \nabla_\theta \log \pi_\theta(a \mid x)$$

- The functions ϕ are called **compatible features**
- If we approximate Q^π using the features ϕ, we recover the gradient

Policy gradient theorem II

Theorem: Given the compatible basis functions

$$\phi(x, a) = \nabla_\theta \log \pi_\theta(a \mid x),$$

let

$$Q_{\boldsymbol{w}}(x, a) = \boldsymbol{w}^\top \phi(x, a),$$

with

$$\boldsymbol{w} = \operatorname{argmin} \mathbb{E}_{\mu_\theta, \pi_\theta} \left[(Q^{\pi_\theta}(\mathrm{x}, \mathrm{a}) - Q_{\boldsymbol{w}}(\mathrm{x}, \mathrm{a}))^2 \right]$$

Then,

$$\nabla_\theta V(\theta) = \mathbb{E}_{\mu_\theta, \pi_\theta} \left[\nabla_\theta \log \pi_\theta(\mathrm{a} \mid \mathrm{x}) Q_{\boldsymbol{w}}(\mathrm{x}, \mathrm{a}) \right].$$

Actor-critic methods

- They provide satisfactory solutions both for the actor and the critic
- Much used in robotics
- Many variations exist:
  - Critic using batch learning
  - Actor using adjusted/natural gradients
  - Multi time scales
  - …

Examples in practice

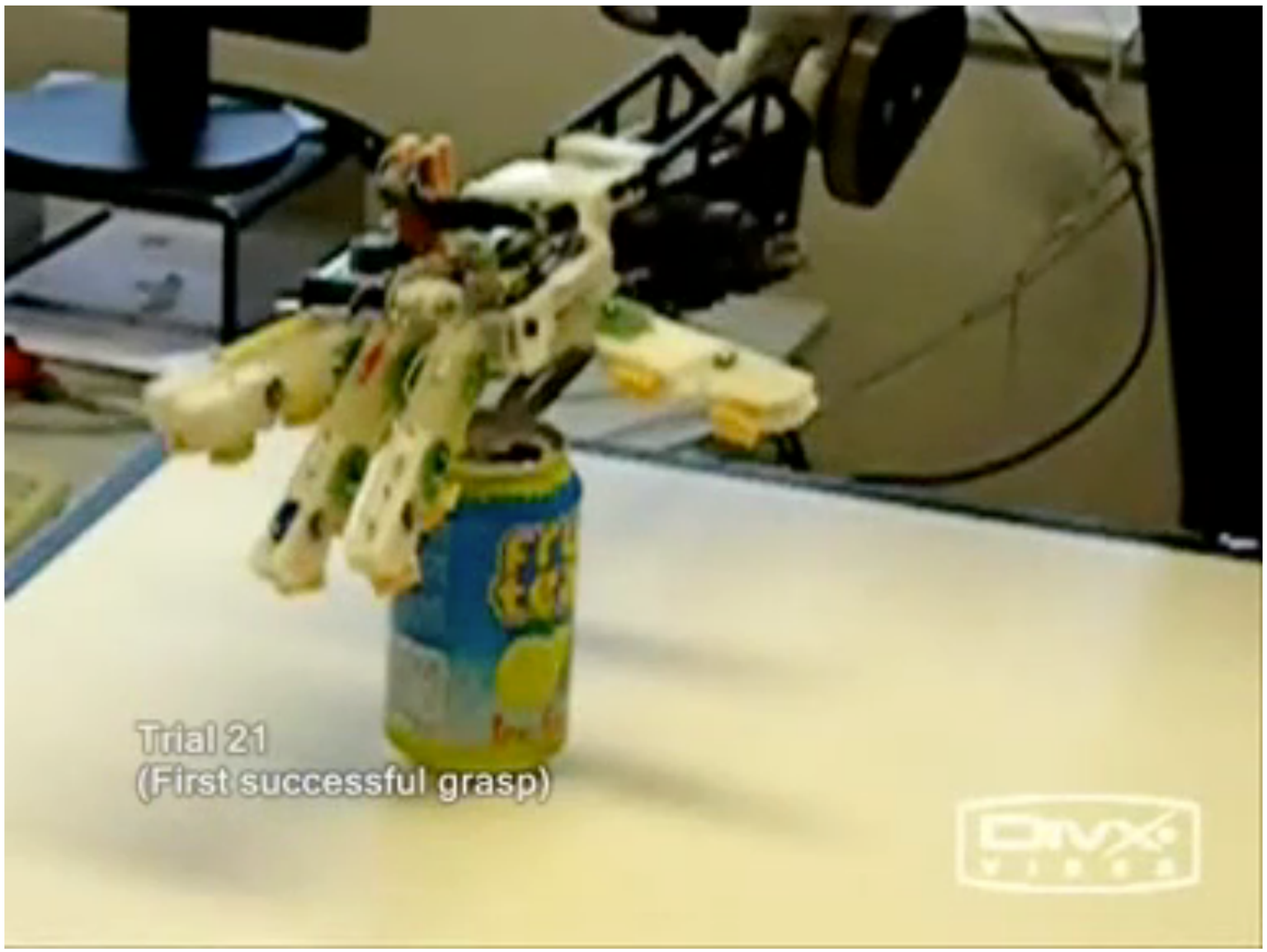
- Gait learning: initial policy

Examples in practice

- Gait learning: final policy

Examples in practice

- Learning to grasp:



Examples in practice

- Learning to intercept a ping-pong ball: