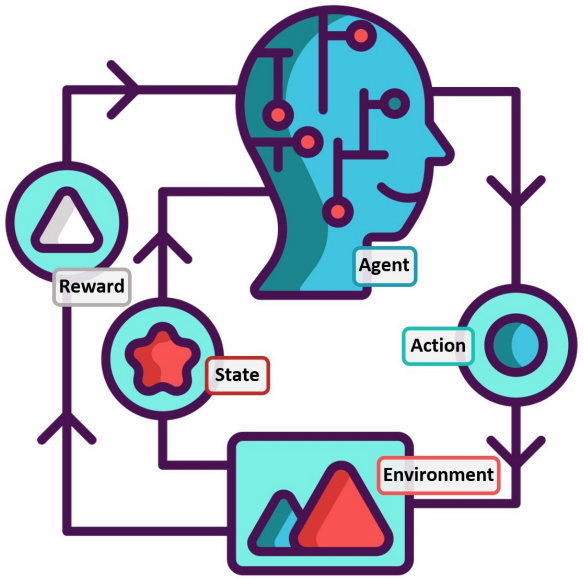
AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

Reinforcement Learning
2025/2026

RL2
Reinforcement Learning Research Lab @ UniPD

Lecture #18
Policy Gradient Methods

Gian Antonio Susto

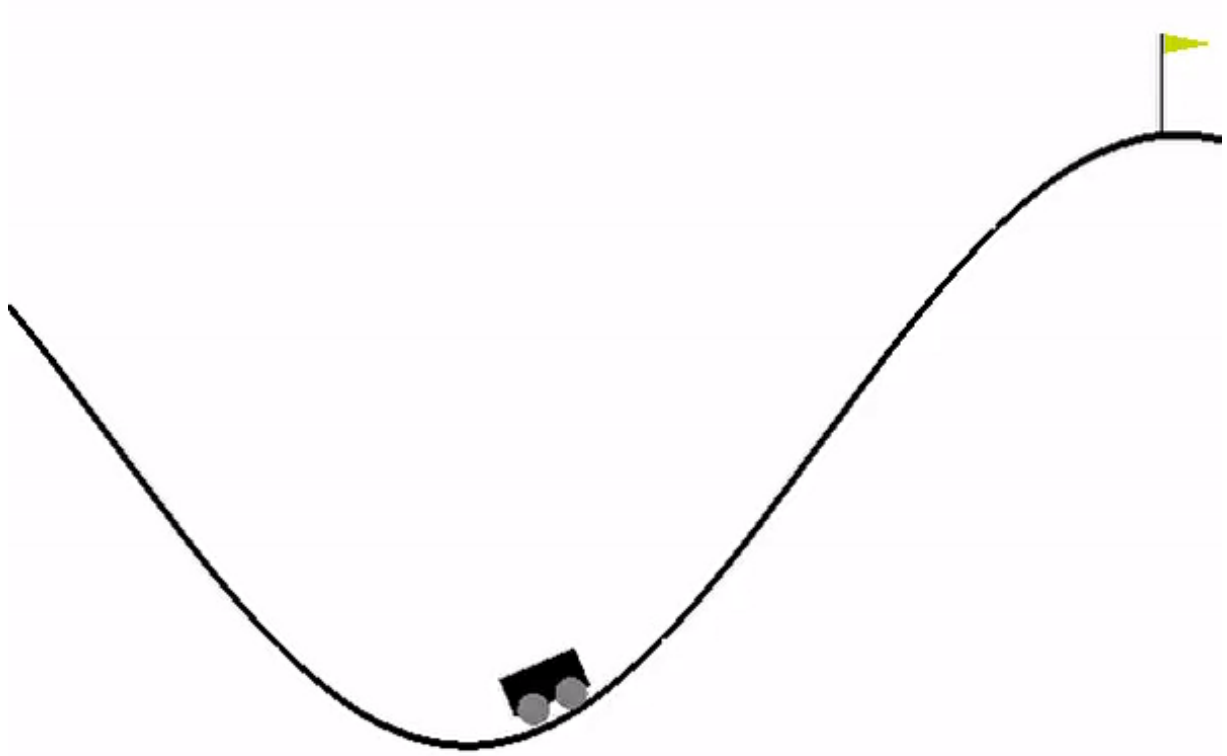


Announcements before starting

- The google form for enrolling in the second partial (19th of December) will be available soon
- For those interested in taking a quick look at their first partial, after the lecture we will have a short session in room Da

Recap – The Mountain car problem

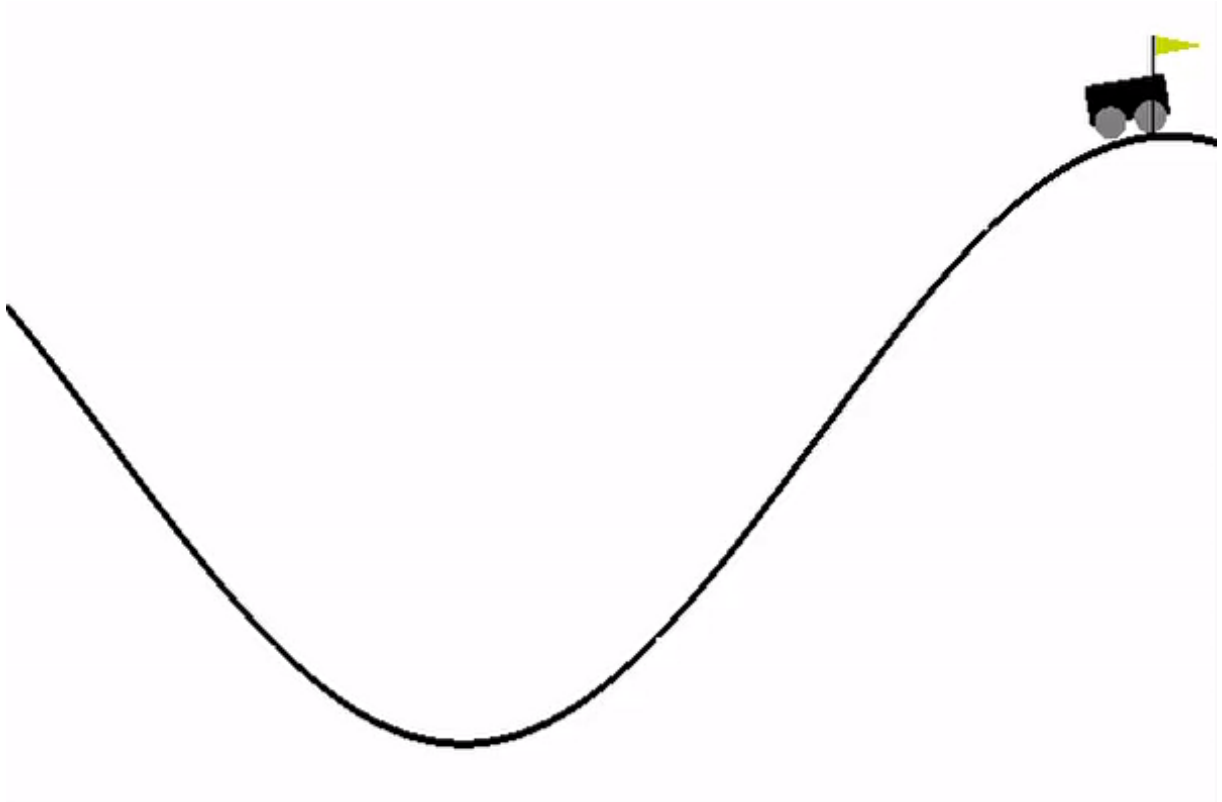
Example: Mountain Car problem (Example 10.1 of the book)



D. Unzueta https://towardsdatascience.com/reinforcement-learning-applied-to-the-mountain-car-problem-1c4fb16729ba

Recap – The Mountain car problem

Example: Mountain Car problem (Example 10.1 of the book)



D. Unzueta https://towardsdatascience.com/reinforcement-learning-applied-to-the-mountain-car-problem-1c4fb16729ba

Recap – The Mountain car problem

Example: Mountain Car problem (Example 10.1 of the book)

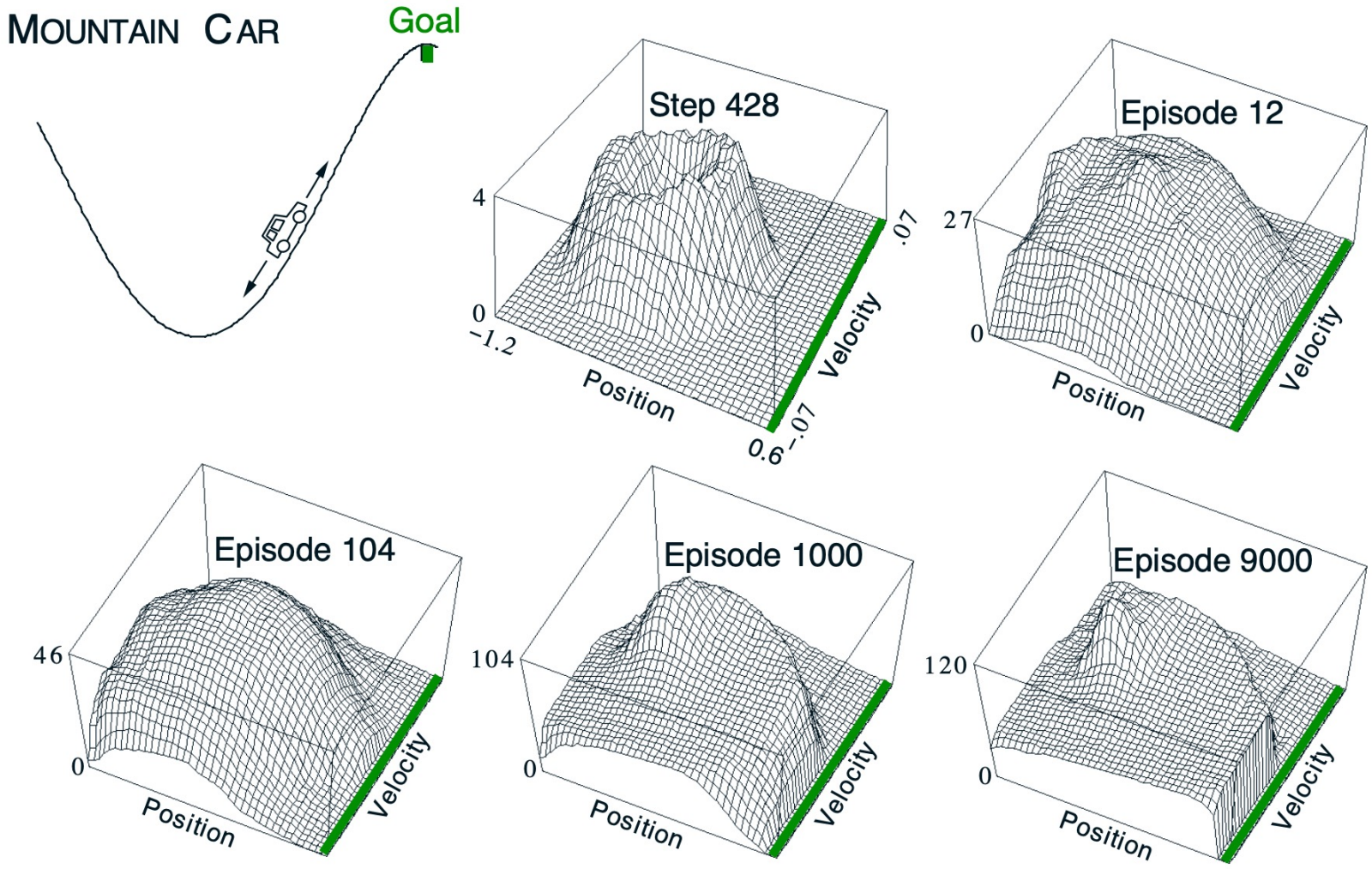


Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function $(-\max_a \hat{q}(s, a, \mathbf{w}))$ learned during one run.

Recap – The Mountain car problem

Example: Mountain Car problem (Example 10.1 of the book)

Any ideas on the form of the optimal policy?

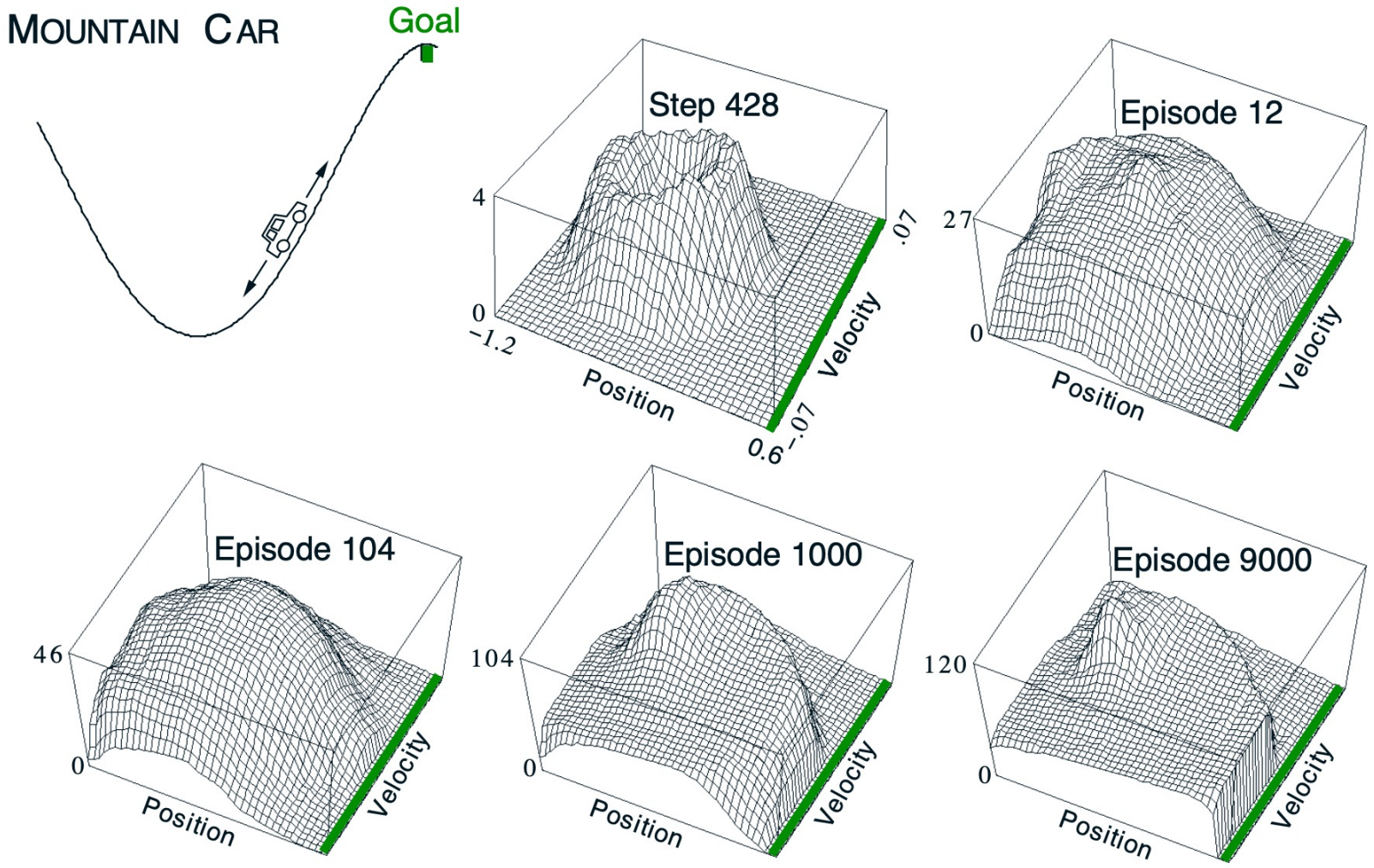
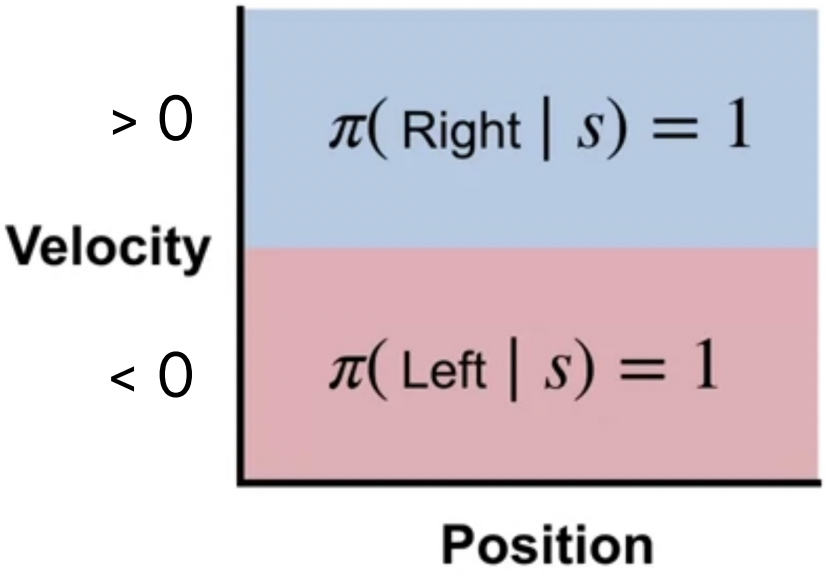


Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function $(-\max_a \hat{q}(s, a, \mathbf{w}))$ learned during one run.

Recap – The Mountain car problem

Example: Mountain Car problem (Example 10.1 of the book)



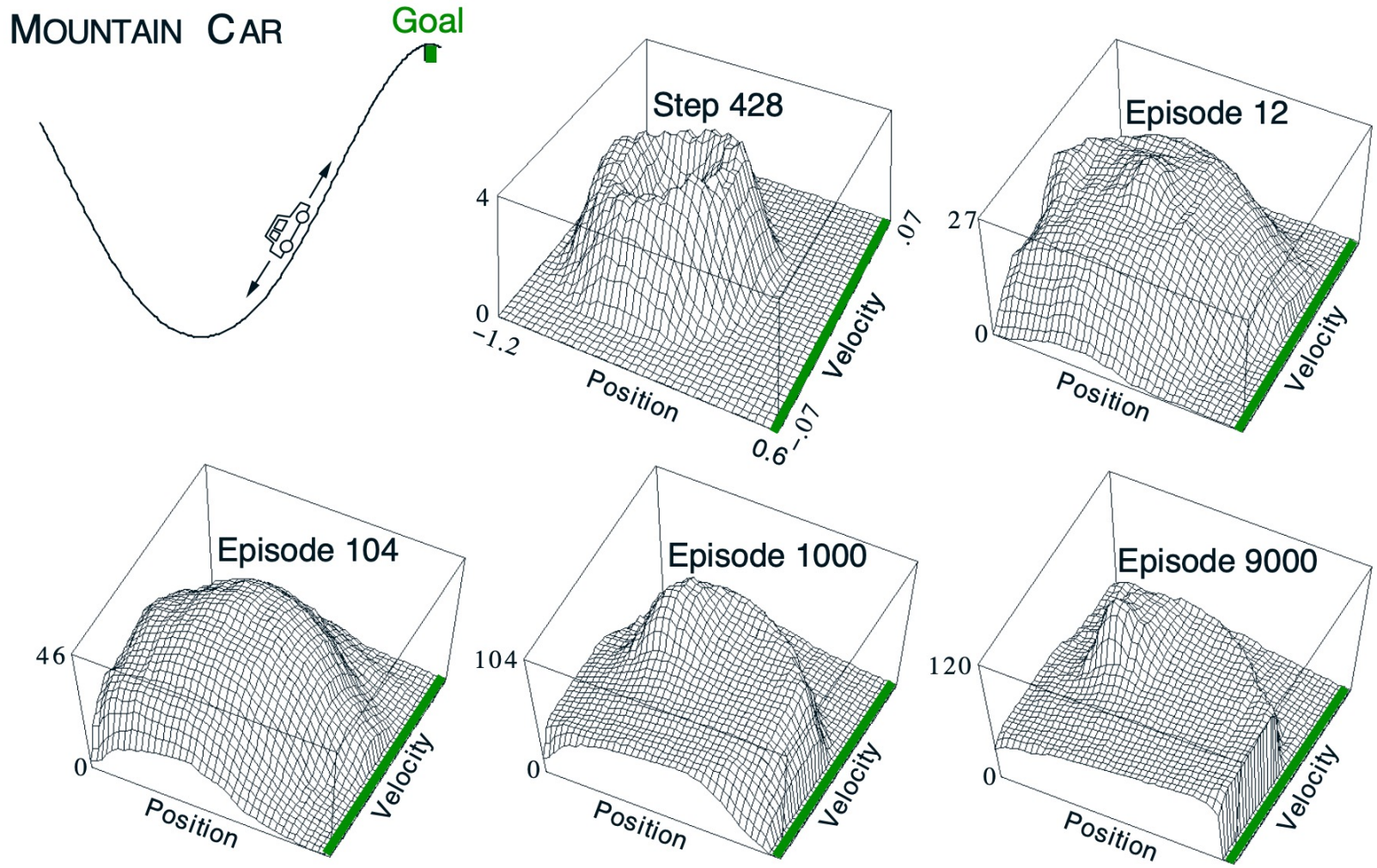


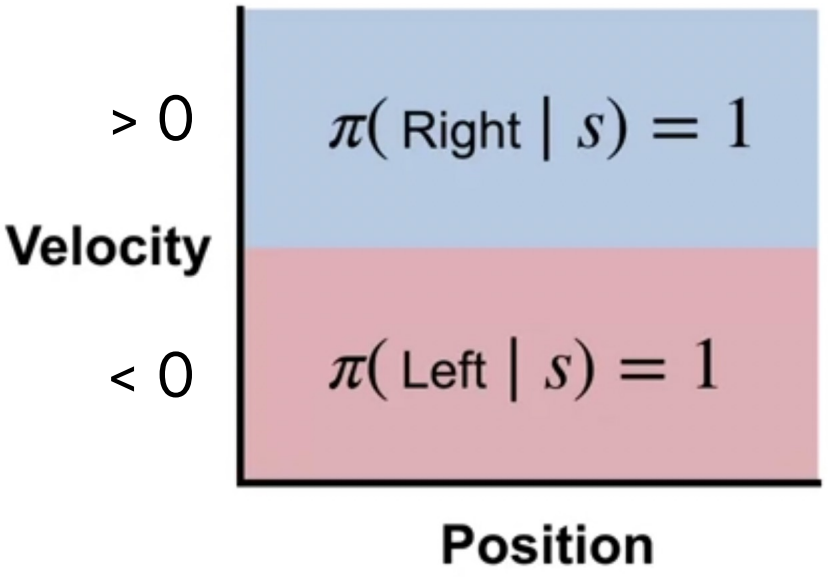
Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function ($-\max_a \hat{q}(s, a, \mathbf{w})$) learned during one run.

Recap – The Mountain car problem

Example: Mountain Car problem

Was the usage of V truly necessary in this case?

the



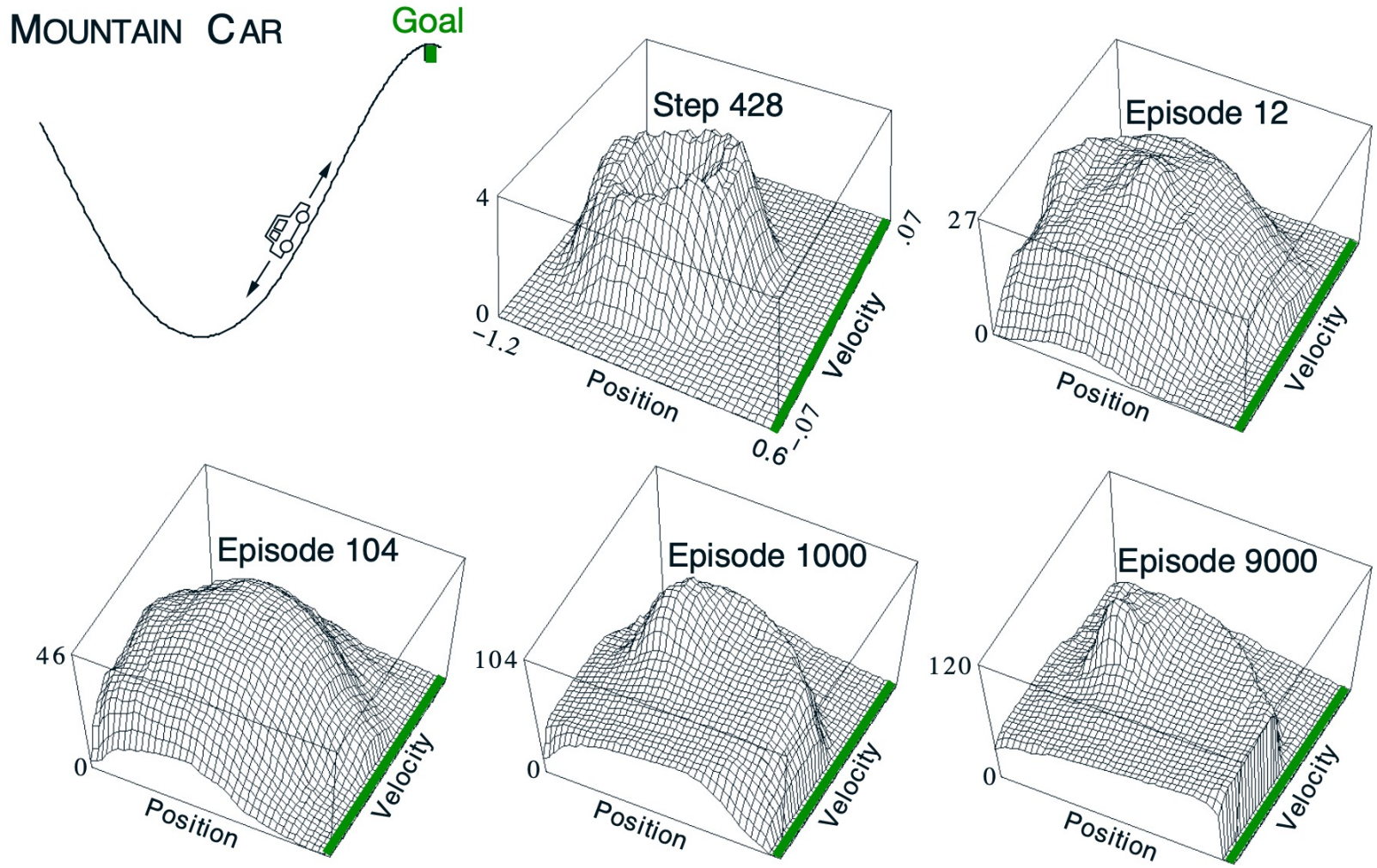
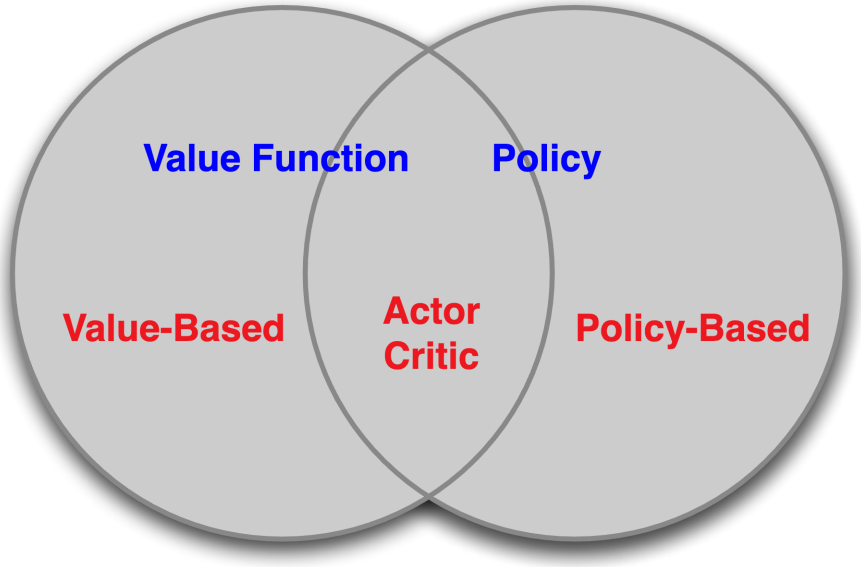


Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function ($-\max_a \hat{q}(s, a, \mathbf{w})$) learned during one run.

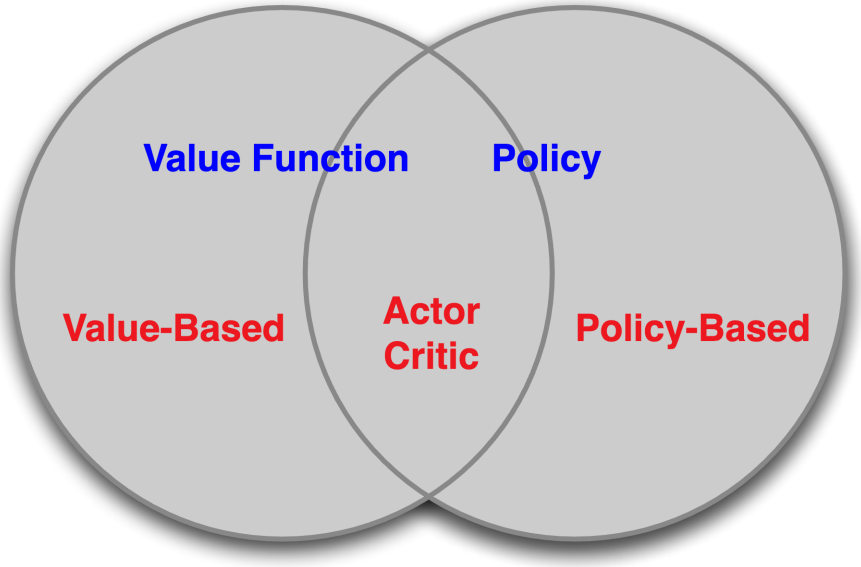
There are RL approaches that do not need a value function!



Up until now we have been dealing with Value-Based approaches!

That are approaches that do not need the definition of a Value function (V and/or Q) and directly act on the policy!

There are RL approaches that do not need a value function!



Up until now we have been dealing with Value-Based approaches!

That are approaches that do not need the definition of a Value function (V and/or Q) and directly act on the policy!

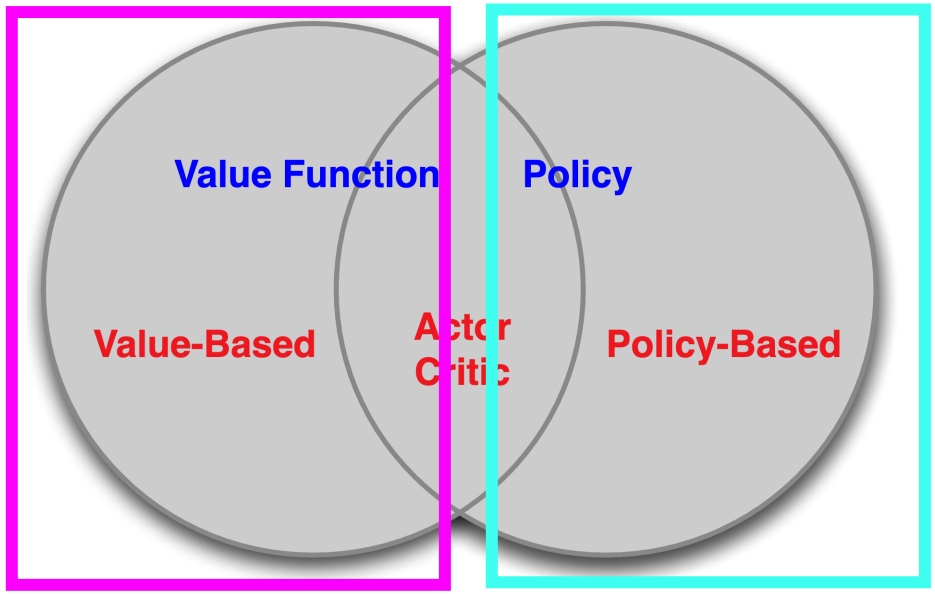
Crude taxonomy of model-free RL approaches:

Value Based
- Learnt Value Function
- Implicit policy(e.g. epsilon-greedy)

Policy Based
- No Value Function
- Learnt Policy

Actor-Critic
- Learnt Value Function
- Learnt Policy

There are RL approaches that do not need a value function!

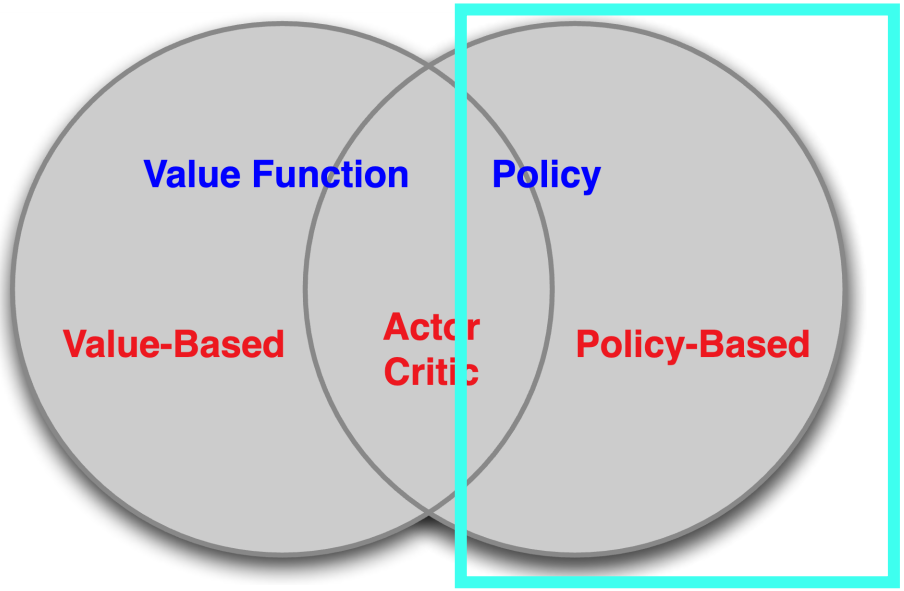


$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

$$\pi_\theta(s,a) = \mathbb{P}[a \mid s, \theta]$$

We will directly parametrize the policy and use data to find 'better parameters' (remember that π still need to be a distribution!)

Policy-based approaches



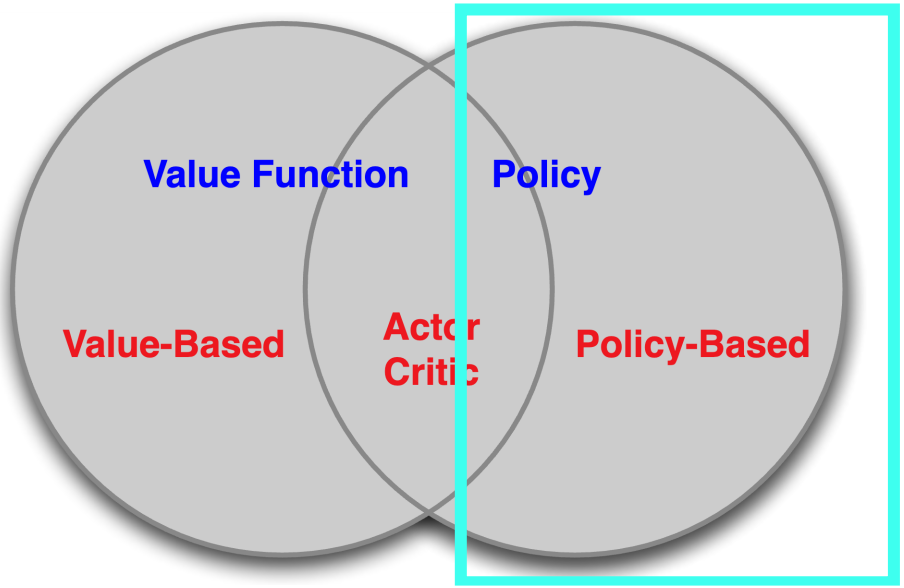
Pros of policy-based approaches:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies (remember, in value-based there is always a ‘max’ operation involved)

Cons:

- Typically converge to a local rather than global optimum
- Evaluating how good a policy is, it is typically inefficient and high variance

Policy-based approaches



Intuition: “With continuous policy parameterization the action probabilities change smoothly as a function of the learned parameter, whereas in epsilon-greedy selection the action probabilities may change dramatically for an arbitrarily small change in the estimated action values, if that change results in a different action having the maximal value.”

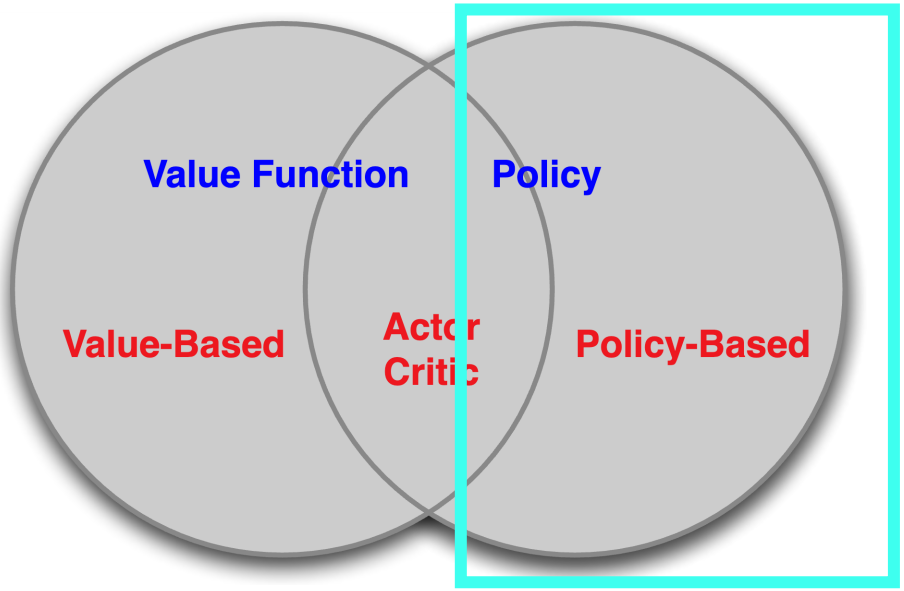
Pros of policy-based approaches:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies (remember, in value-based there is always a ‘max’ operation involved)

Cons:

- Typically converge to a local rather than global optimum
- Evaluating how good a policy is, it is typically inefficient and high variance

Policy-based approaches



Why a stochastic policy?

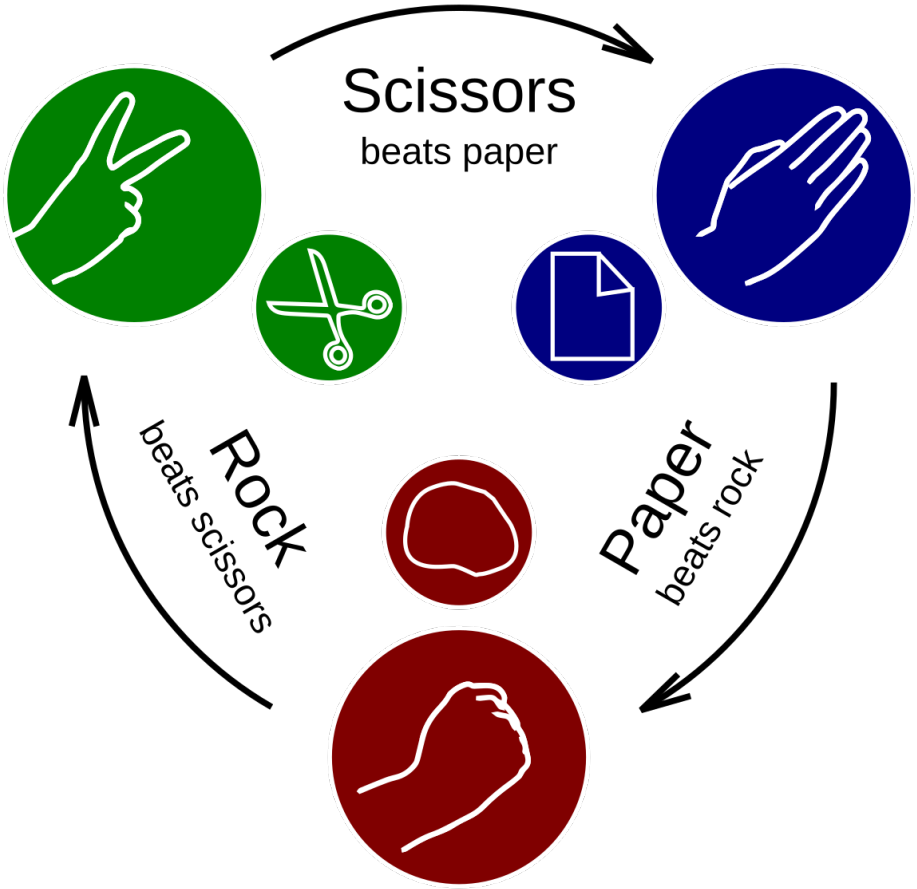
Pros of policy-based approaches:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies (remember, in value-based there is always a ‘max’ operation involved)

Cons:

- Typically converge to a local rather than global optimum
- Evaluating how good a policy is, it is typically inefficient and high variance

Example of a desirable stochastic policy #01: Rock-Paper-Scissors



In the game of rock-paper-scissors it can be proved that the optimal policy is a uniform random policy!*

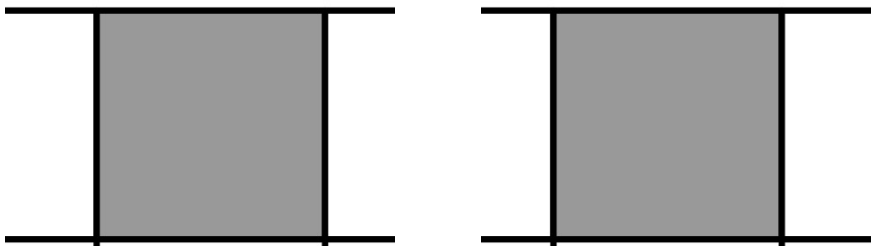
If something is done deterministically, the opponent can exploit the deterministic choice/pattern.

* In terms of Nash equilibrium, ie the game-theory concept of optimality

Example of a desirable stochastic policy #02: ‘Aliased Gridworld’

Stochastic approaches are desirable in partially observable settings and/or when the ‘features’ describing the state provide incomplete information

Example of a desirable stochastic policy #02: ‘Aliased Gridworld’



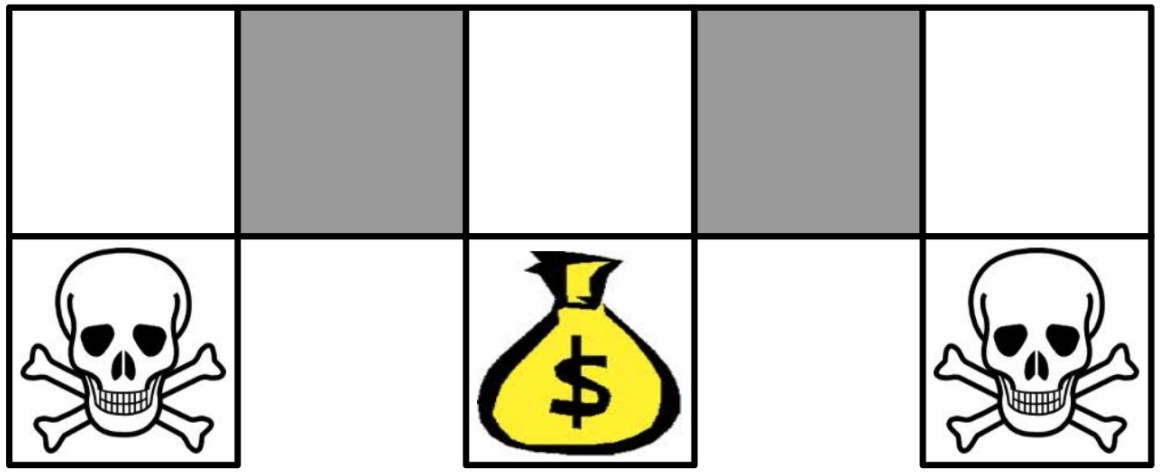
Stochastic approaches are desirable in partially observable settings and/or when the ‘features’ describing the state provide incomplete information

In this small gridworld, we consider a feature-based representation of this form (for all N, E, S, W)

$$\phi(s, a) = \mathbf{1}(\text{wall to N}, a = \text{move E})$$

The agent cannot differentiate the grey states!

Example of a desirable stochastic policy #02: ‘Aliased Gridworld’



A deterministic vs a stochastic policy how they will behave?

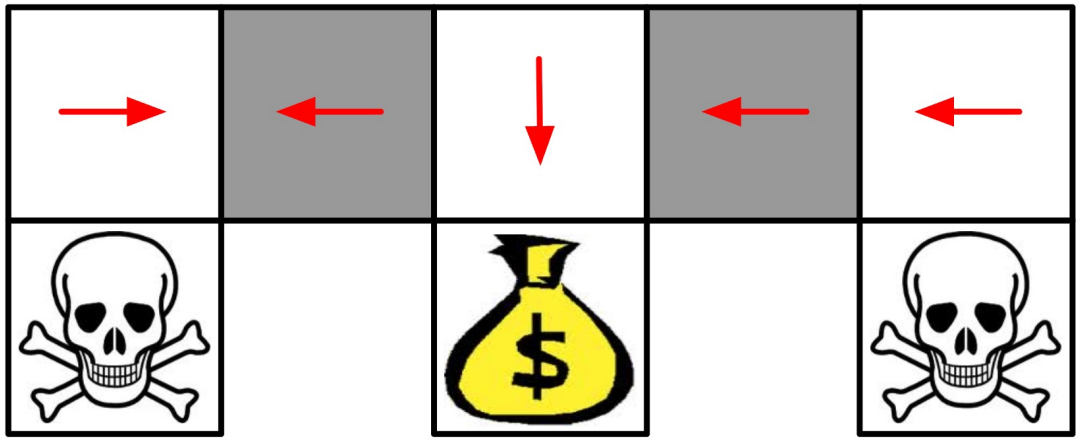
In this small gridworld, we consider a feature-based representation of this form (for all N, E, S, W)

$$\phi(s, a) = \mathbf{1}(\text{wall to N}, a = \text{move E})$$

The agent cannot differentiate the grey states

Example of a desirable stochastic policy #02: ‘Aliased Gridworld’

Example of optimal deterministic policy



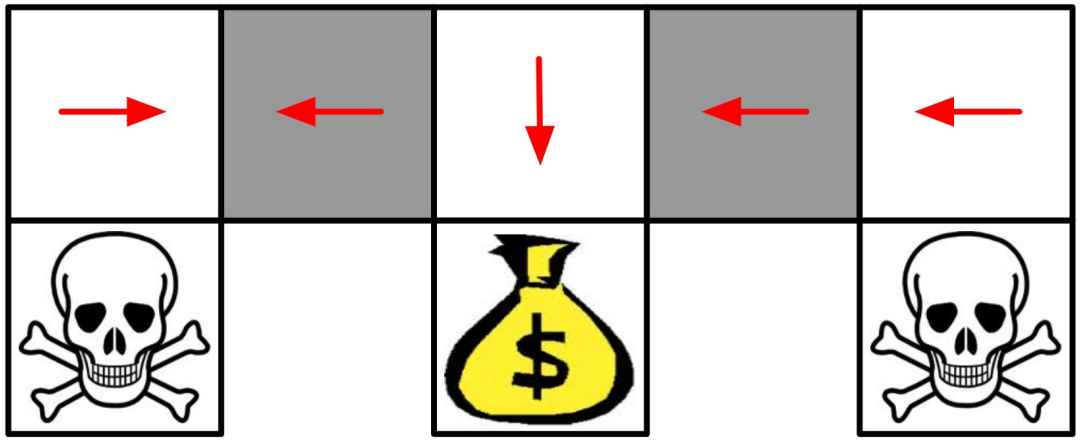
An optimal deterministic policy will either

- move W in both grey states (shown by red arrows)
- move E in both grey states

Either way, it can get stuck and never reach the money

Example of a desirable stochastic policy #02: ‘Aliased Gridworld’

Example of optimal deterministic policy

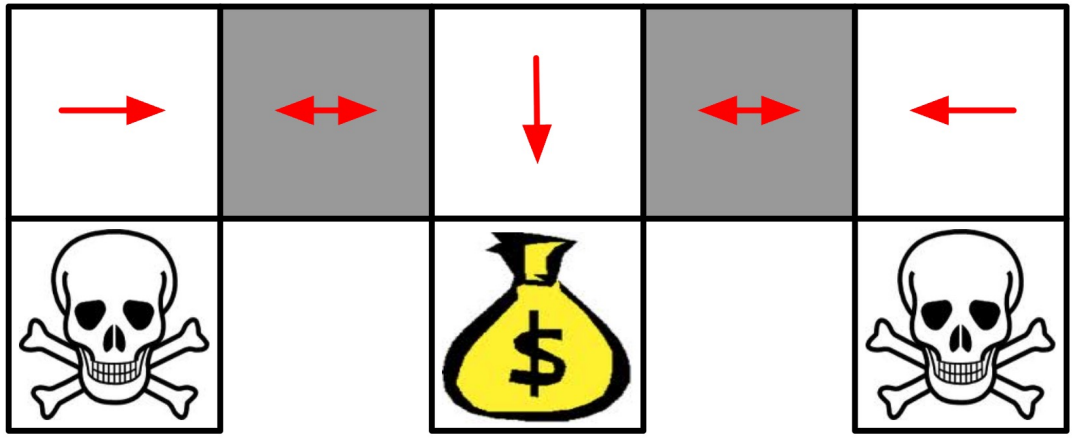


An optimal deterministic policy will either

- move W in both grey states (shown by red arrows)
- move E in both grey states

Either way, it can get stuck and never reach the money

Example of optimal stochastic policy



An optimal stochastic policy in the grey states

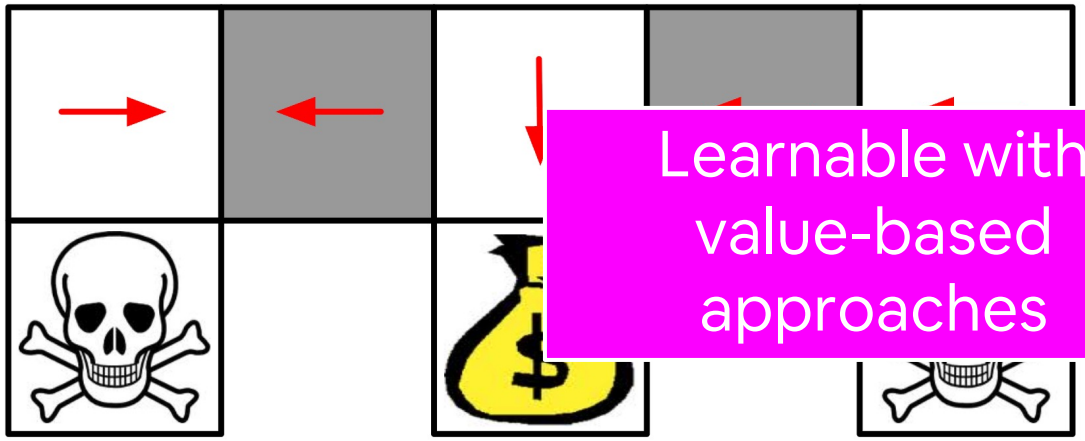
$$\pi_\theta(\text{wall to N and S, move E}) = 0.5$$

$$\pi_\theta(\text{wall to N and S, move W}) = 0.5$$

It will reach the goal state in a few steps with high probability

Example of a desirable stochastic policy #02: 'Aliased Gridworld'

Example of optimal deterministic policy



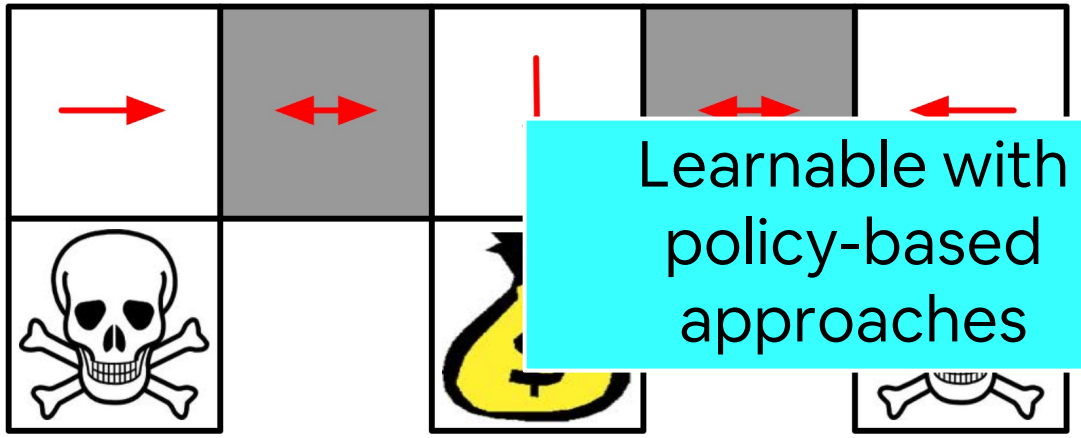
Learnable with value-based approaches

An optimal deterministic policy will either

- move W in both grey states (shown by red arrows)
- move E in both grey states

Either way, it can get stuck and never reach the money

Example of optimal stochastic policy



Learnable with policy-based approaches

An optimal stochastic policy in the grey states

$$\pi_\theta(\text{wall to N and S, move E}) = 0.5$$

$$\pi_\theta(\text{wall to N and S, move W}) = 0.5$$

It will reach the goal state in a few steps with high probability

Don’t get confused!

- In MDPs, the optimality theorem holds: ‘there is always an optimal deterministic policy’
- In partially observable MDPs or in cases where the used features in a VFA case do not fully capture the problem, stochastic policies can be optimal!

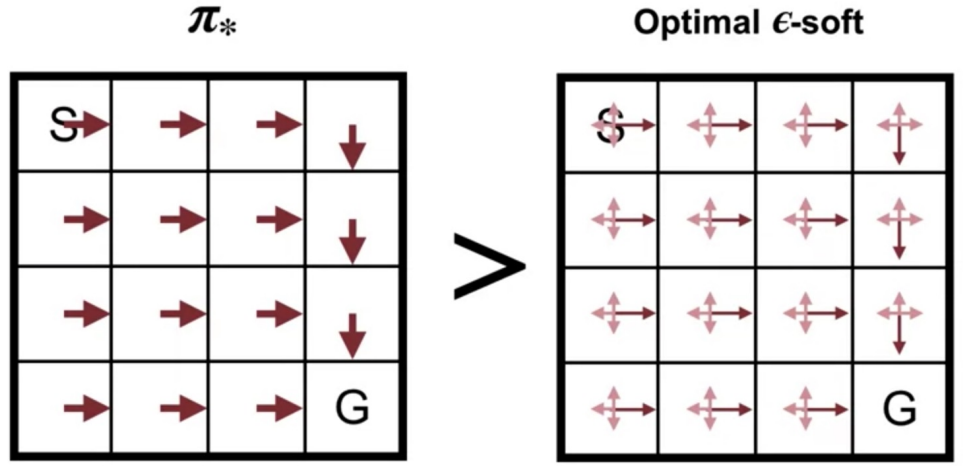
Fully observable vs partially observable MDP

Don’t get confused!

- In MDPs, the optimality theorem holds: ‘there is always an optimal deterministic policy’
- In partially observable MDPs or in cases where the used features in a VFA case do not fully capture the problem, stochastic policies can be optimal!

- We can achieve stochastic policies also with value-based approaches, but as discussed, they are typically not optimal!

Fully observable vs partially observable MDP



A typical Policy parametrization

- We consider that the policy is differentiable with respect to its parameters θ
- The derivative $\nabla_{\boldsymbol{\theta}} \pi(a|s, \boldsymbol{\theta})$ exists and it is always finite
- An example of policy function, the exponential softmax distribution ->

$$\pi_{\theta}(s, a) = \mathbb{P}[a \mid s, \theta]$$

$$\pi(a|s, \boldsymbol{\theta}) = \frac{\exp(h(s, a, \boldsymbol{\theta}))}{\sum_b \exp(h(s, b, \boldsymbol{\theta}))}$$

where h() are parametrized numerical preferences

(similarly to what we have seen for example in Gradient Bandits)

Policy Objective Functions

- How to find the 'best' parameters for a policy?

$$\pi_\theta(s, a) = \mathbb{P}[a \mid s, \theta]$$

- We need an objective function J! A mathematical description of what 'best' is!

- Any ideas?

Policy Objective Functions: Examples

- Start value (for episodic environment)

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- Average value (continuous)

$$J_{avV}(\theta) = \sum_s \mu(s) V^{\pi_\theta}(s)$$

- Average reward per time-step

$$J_{avR}(\theta) = \sum_s \mu(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

$\mu(s)$: distribution associated with the probability of being in state for an MDP given a policy (not always easy to have)

Policy Objective Functions: Examples

- Start value (for episodic environment)

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- Average value (continuous)

$$J_{avV}(\theta) = \sum_s \mu(s) V^{\pi_\theta}(s)$$

- Average reward per time-step

$$J_{avR}(\theta) = \sum_s \mu(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

$\mu(s)$: distribution associated with the probability of being in state for an MDP given a policy (not always easy to have)

The same approach (policy-gradient) applies to all choices of J

Policy Objective Functions: Examples

- Start value (for episodic environment)

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- Average value (continuous)

$$J_{avV}(\theta) = \sum_s \mu(s) V^{\pi_\theta}(s)$$

- Average reward per time-step

$$J_{avR}(\theta) = \sum_s \mu(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

We are looking for parameters to increase J! RL is typically about maximization, not minimization

iated with the
g in state for an MDP
given a policy (not always easy to have)

The same approach (policy-gradient) applies to all choices of *J*

Policy Objective Functions: E

This is the true V (an ideal/theoretic scenario)

- Start value (for episodic environment)

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- Average value (continuous)

$$J_{avV}(\theta) = \sum_s \mu(s) V^{\pi_\theta}(s)$$

- Average reward per time-step

$$J_{avR}(\theta) = \sum_s \mu(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

We are looking for parameters to increase J! RL is typically about maximization, not minimization

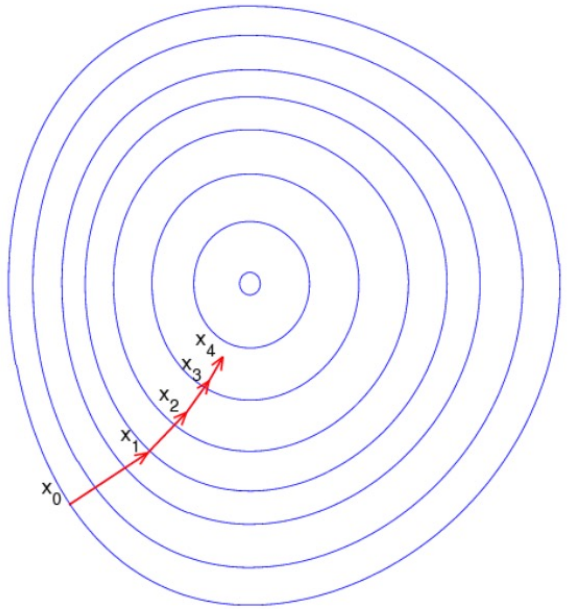
iated with the
g in state for an MDP
given a policy (not always easy to have)

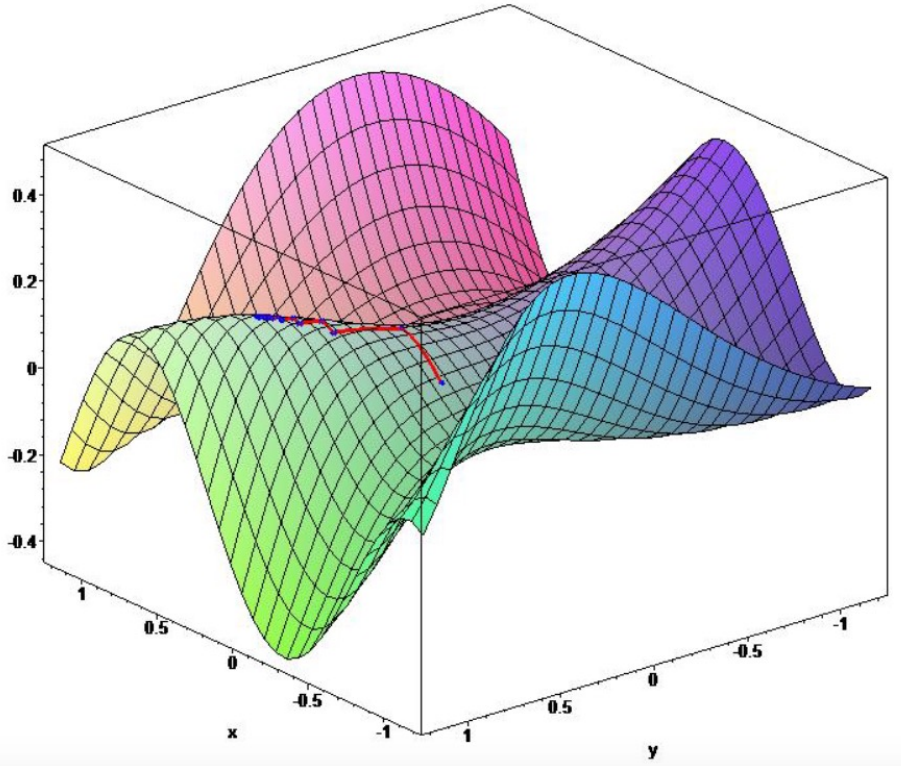
The same approach (policy-gradient) applies to all choices of *J*

Policy Gradient

- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy w.r.t. parameters θ

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$





$$\theta_{t+1} = \theta_t + \alpha\Delta\theta$$

Step-size

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

Policy gradient

Computing the gradient with finite differences

For each dimensions k in [1, n]:

1. We estimate the k-th partial derivative of the objective function w.r.t. theta
2. We perturbate theta by small amount in k-th dimension

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is unit vector with 1 in kth component, 0 elsewhere

Computing the gradient with finite differences

For each dimensions k in [1, n]:

1. We estimate the k-th partial derivative of the objective function w.r.t. theta
2. We perturbate theta by small amount in k-th dimension

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is unit vector with 1 in kth component, 0 elsewhere

- Use *n* evaluations to compute policy gradient in *n* dimensions
- This is a highly data/experiment expensive approach
- Works for arbitrary policies, even if policy is not differentiable

Computing the gradient with finite differences

For each dimensions k in [1, n]:

1. We estimate the k-th partial derivative of the objective function w.r.t. theta
2. We perturbate theta by small amount in k-th dimension

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is unit vector with 1 in kth component, 0 elsewhere

- Use *n* evaluations to compute policy gradient in *n* dimensions
- This is a highly data/experiment expensive approach
- Works for arbitrary policies, even if policy is not differentiable

We would like to have an analytical way to compute the gradient!

The Policy Gradient Theorem (sec. 13.2)

It can be shown – in the episodic case – that (see if curious the proof in the book)

$$\nabla J(\boldsymbol{\theta}) \propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

The Policy Gradient Theorem (sec. 13.2)

It can be shown – in the episodic case – that (see if curious the proof in the book)

$$\nabla J(\boldsymbol{\theta}) \propto \sum_{s} \mu(s) \sum_{a} q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

Unfortunately, in stochastic gradient ascent this is not directly usable: we need a way to obtain samples (something that I can get from experience) such that the expectation of the sample gradient is proportional to the actual gradient of the performance measure as a function of the parameter

$J(\theta)$: performance measure

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$ gradient ascent

Towards REINFORCE: Monte Carlo Policy Gradient

We are summing over all the possible state the probability of actually being in that state given the policy -> expectation

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}),$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$(\text{because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))$$

Towards REINFORCE: Monte Carlo Policy Gradient

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}),$$

Dividing and multiplying by the policy

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$(\text{because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))$$

Towards REINFORCE: Monte Carlo Policy Gradient

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}),$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$(\text{because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))$$

Replacing a by the sample A_t following policy π

Towards REINFORCE: Monte Carlo Policy Gradient

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}),$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$(\text{because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))$$

By definition

Towards REINFORCE: Monte Carlo Policy Gradient

We got rid of q (and mu) and we replace it with data coming from experience!

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}),$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_\pi \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$

$$(\text{because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))$$

The REINFORCE update (A Monte Carlo approach)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}$$

The REINFORCE update (A Monte Carlo approach)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}$$

The update increases the parameter vector in a direction:

(i) proportional to the return - makes sense! Because it causes the parameter to move most in the directions that favour actions that yield the highest return

(ii) inversely proportional to the action probability. This makes sense as well: otherwise, actions that are selected frequently are at an advantage (the updates will be more often in their direction) and might win out even if they do not yield the highest return.

The REINFORCE update (A Monte Carlo approach)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}$$

$$\doteq \boldsymbol{\theta}_t + \alpha G_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}_t)$$

Score function

The REINFORCE algorithm (A Monte Carlo approach)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$

Algorithm parameter: step size $\alpha > 0$ Hyperparameter

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$

Loop for each step of the episode $t = 0, 1, \ldots, T-1$:

$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

The REINFORCE algorithm (A Monte Carlo approach)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Initialization

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$

Loop for each step of the episode $t = 0, 1, \ldots, T-1$:

$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

The REINFORCE algorithm (A Monte Carlo approach)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Monte Carlo: 1 Loop!

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$

Loop for each step of the episode $t = 0, 1, \ldots, T-1$:

$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

The REINFORCE algorithm (A Monte Carlo approach)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Going backward for efficiency -> every visit version!

The REINFORCE algorithm (A Monte Carlo approach)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

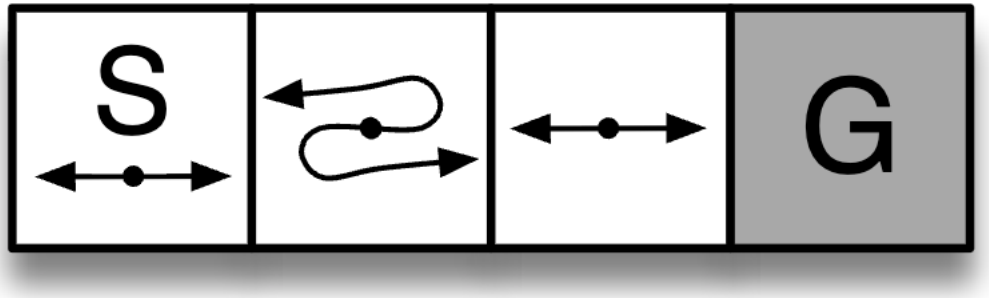
Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

No V or Q is involved!

Example: short corridor with switched actions

Consider the small corridor gridworld ->

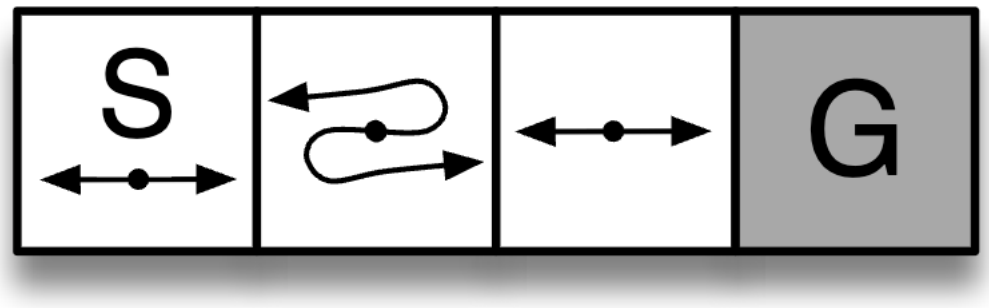
- The reward is -1 per step, as usual
- In each of the three nonterminal states there are only two actions, right and left (but in the second state they are reversed, so that right moves to the left and left moves to the right).
- The problem is difficult because all the states appear identical under the function approximation (we define $\mathbf{x}(s,right)$ = [1,0] and $\mathbf{x}(s, left)$ = [0,1] for all s.

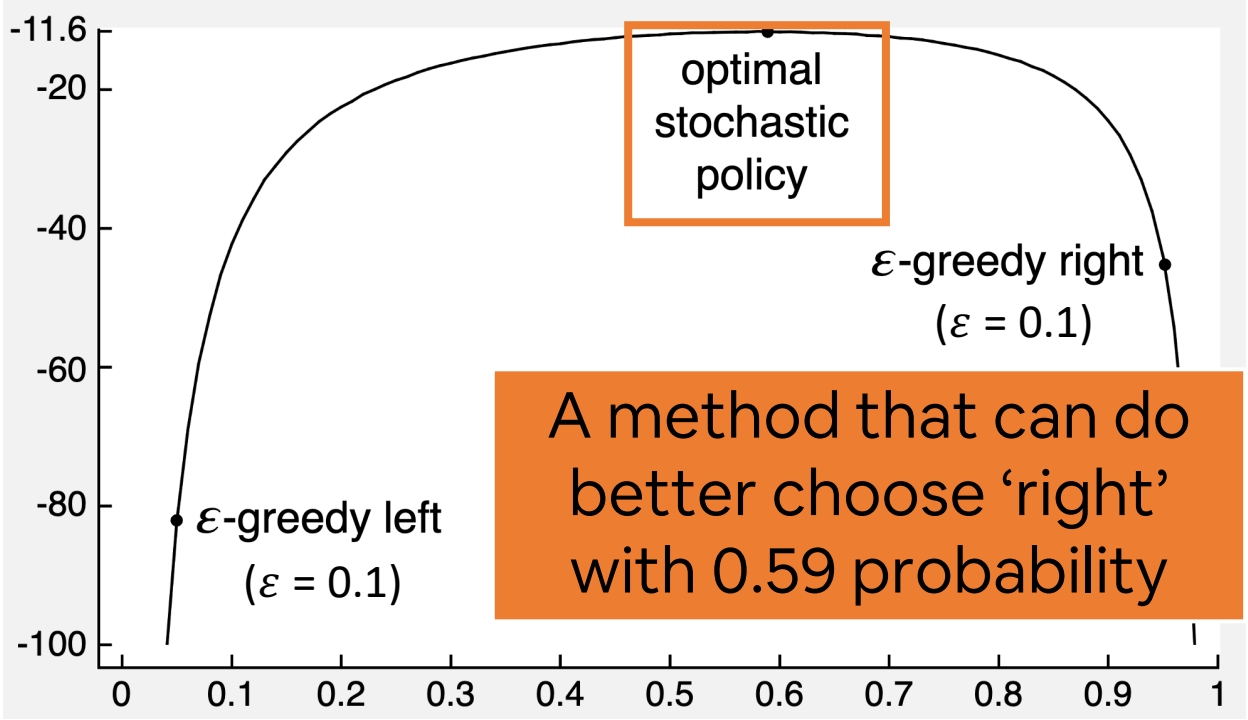


Example: short corridor with switched actions

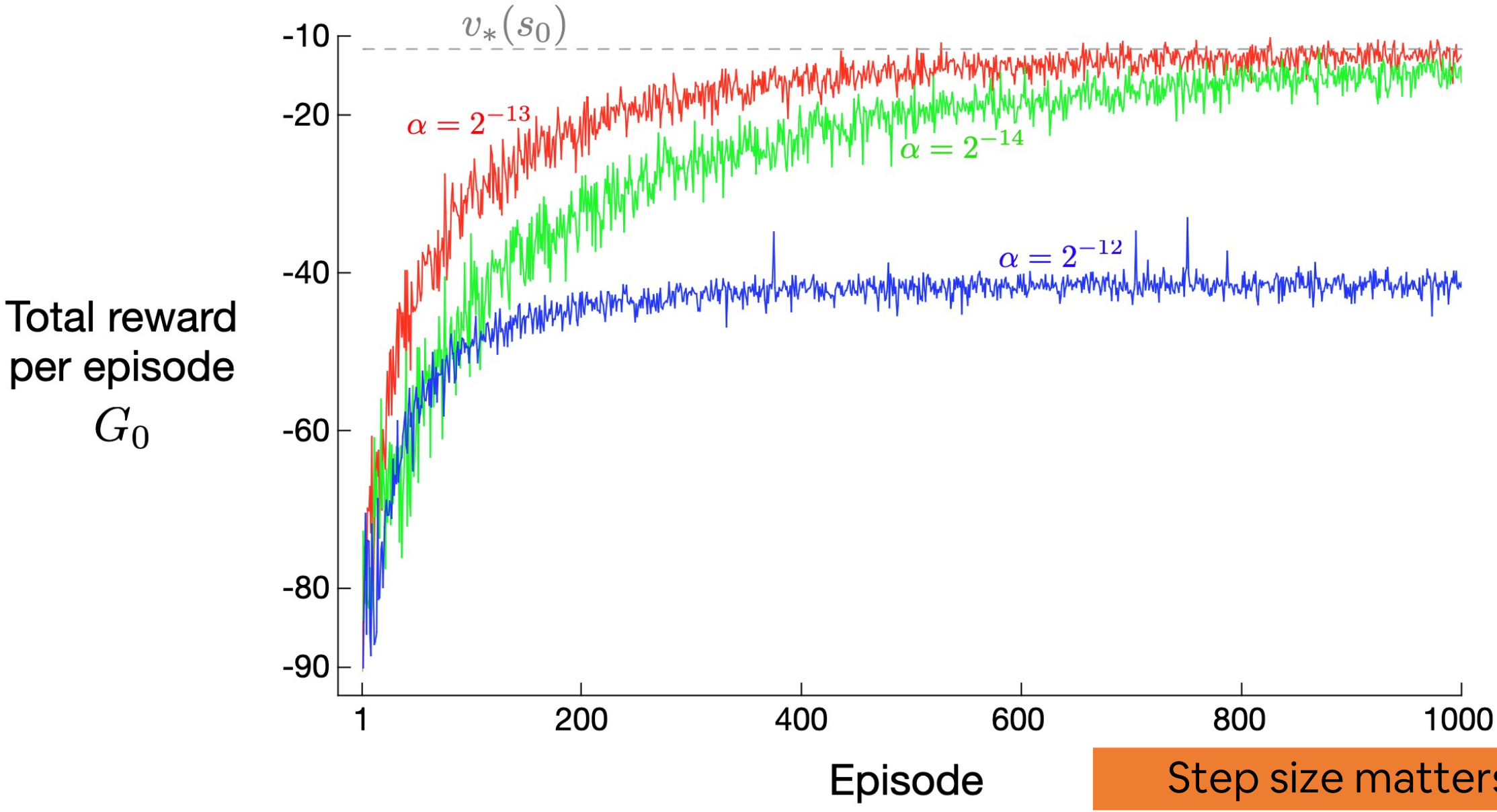
Consider the small corridor gridworld ->

- The reward is -1 per step, as usual
- In each of the three nonterminal states there are only two actions, right and left (but in the second state they are reversed, so that right moves to the left and left moves to the right).
- The problem is difficult because all the states appear identical under the function approximation (we define $\mathbf{x}(s,right)$ = [1,0] and $\mathbf{x}(s, left)$ = [0,1] for all s.
- An epsilon-greedy approach is forced to choose between just two policies: choosing right with high probability $(1 - \varepsilon/2)$ or left with the same high probability





Example: short corridor with switched actions



Step size matters!

REINFORCE – whitening

As all Monte Carlo approaches, also REINFORCE is affected by high variance, leading to

- Unstable learning
- Slow convergence

A solution is the so-called whitening, ie. normalize the returns of each step of the episode by subtracting the mean and dividing by the standard deviation of returns at all time steps within the episode

$$G_t^* = \frac{G_t - \bar{G}}{\sigma_G}$$

$$\theta_{t+1} = \theta_t + \alpha\, G_t^*\, \nabla \ln \pi(A_t|S_t, \theta)$$

REINFORCE with baseline

A different approach is to include an arbitrary baseline b(s)

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(G_t - b(S_t)\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

That can be several choices for the baseline, any ideas?

REINFORCE with baseline

A different approach is to include an arbitrary baseline b(s)

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(G_t - b(S_t)\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

That can be several choices for the baseline, any ideas?

Let's use the estimation of the state value v(s) with one of the method seen previously in the course! (For example, Monte Carlo for simplicity)

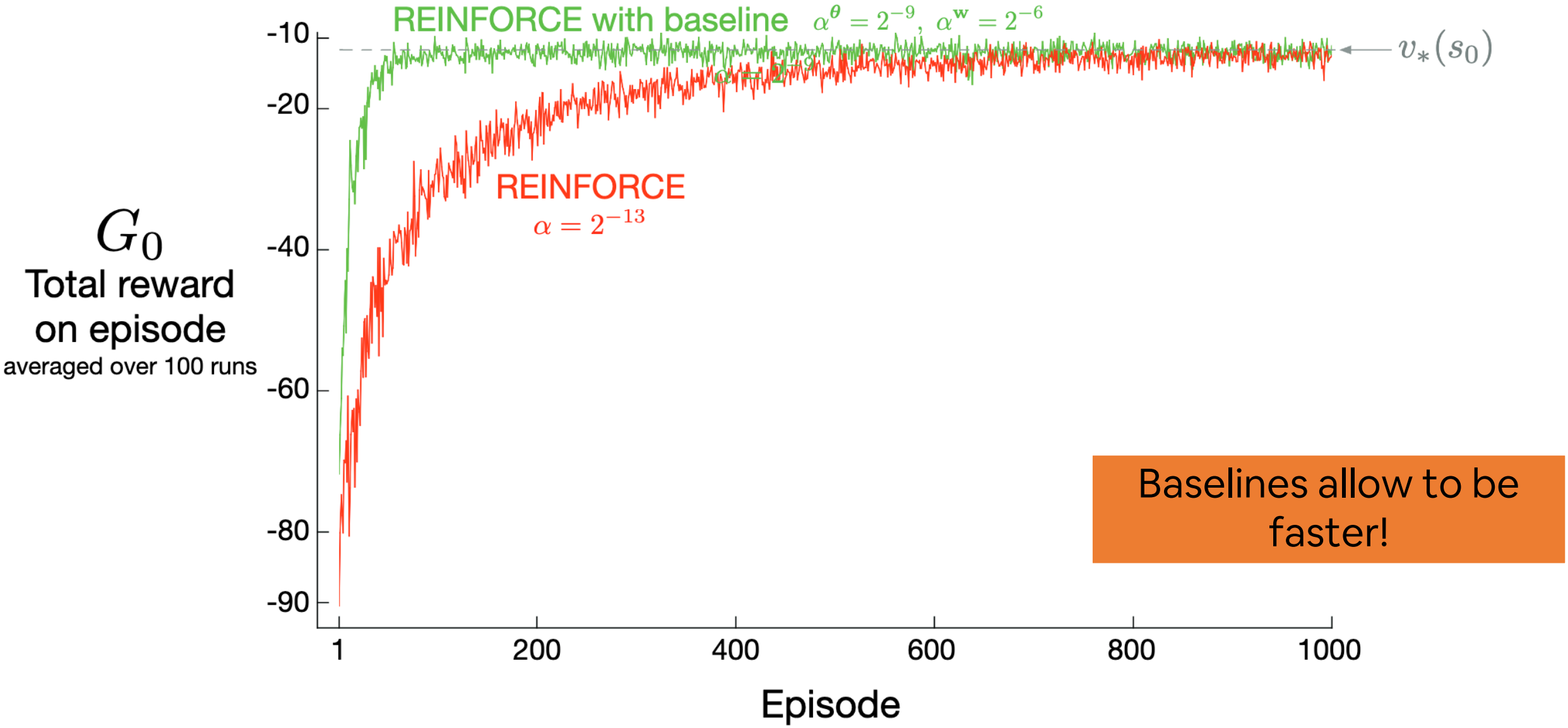
REINFORCE with baseline

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
        $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Example: short corridor with switched actions



Baselines allow to be faster!

Thank you!
Questions?

Lecture #18: Policy Gradient Methods

Gian Antonio Susto