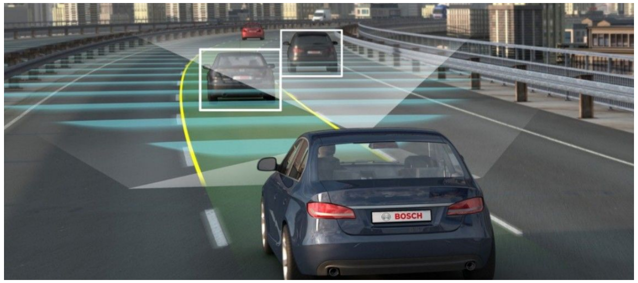
Continuous Reinforcement Learning & Policy Search

Course: Reinforcement Learning, Leiden University

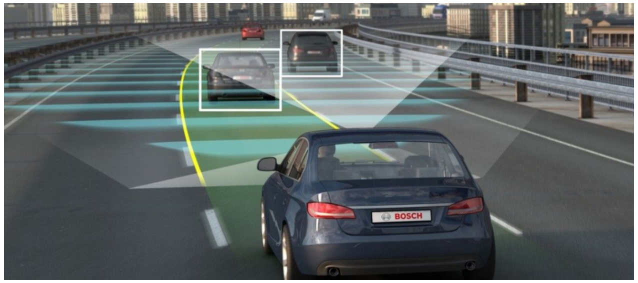
Lecturer: Thomas Moerland

Sequential decision making

Sequential decision making



Sequential decision making



Topic of today!

Content

A. Continuous reinforcement learning

1. Recap: sets, functions, probability
2. Continuous Markov Decision Process
3. Representing the solution

Break

Content

A. Continuous reinforcement learning

1. Recap: sets, functions, probability
2. Continuous Markov Decision Process
3. Representing the solution

Break

B. Policy search

4. Policy gradients
5. Actor-critic
6. Gradient-free policy search

1. Recap: sets, functions, probability

Discrete versus continuous spaces

Discrete set/space → **Continuous set/space**

Discrete versus continuous spaces

Discrete set/space → **Continuous set/space**

Countable elements

Interval

Discrete versus continuous spaces

Discrete set/space

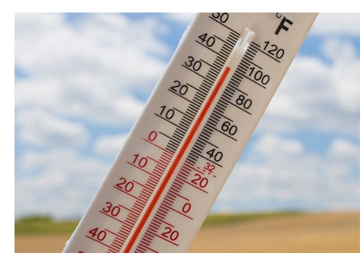
Countable elements

- $\mathcal{X} = \{1, 2, .., n\}$
- $\mathcal{X} = \{\text{up,down,left,right}\}$
- $\mathcal{X} = \{0, 1\}^d$

Continuous set/space

Interval

- $\mathcal{X} = [2, 11]$
- $\mathcal{X} = \mathbb{R}$
- $\mathcal{X} = [0, 1]^d$



Discrete versus continuous functions

$$f : \mathcal{X} \to \mathcal{Y}$$

Discrete versus continuous functions

$$f : \mathcal{X} \to \mathcal{Y}$$

A function f

Domain:

Set in which
the function
takes its input

Range:

Set in which
the function
takes its output

Discrete versus continuous functions

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

A function f

Domain:

Set in which
the function
takes its input

Range:

Set in which
the function
takes its output

Can both be either discrete or continuous!

Discrete versus continuous probability

Discrete probability distribution

Continuous probability distribution

Discrete versus continuous probability

Discrete probability distribution

- Parameters: p_1, p_2,.., p_{n-1}

Continuous probability distribution

- Parameters: depends on distribution, e.g. normal distribution μ,σ

Discrete versus continuous probability

Discrete probability distribution

- Parameters: $p_1, p_2,.., p_{n-1}$

- Probability mass:

$p(X=1)$	$p(X=2)$	$p(X=3)$
0.2	0.4	0.4

Continuous probability distribution

- Parameters: depends on distribution, e.g. normal distribution μ,σ

- Probability density:

e.g. $$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big)$$

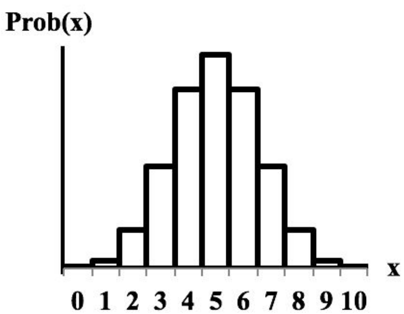
Discrete versus continuous probability

Discrete probability distribution

- Parameters: $p_1, p_2, .., p_{n-1}$

- Probability mass:

$p(X=1)$	$p(X=2)$	$p(X=3)$
0.2	0.4	0.4

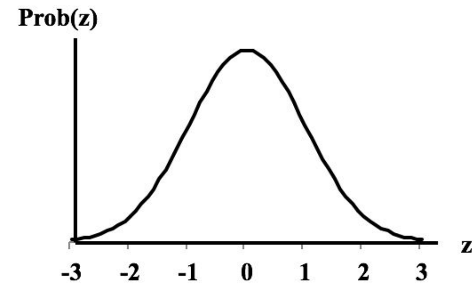


Continuous probability distribution

- Parameters: depends on distribution, e.g. normal distribution μ,σ

- Probability density:

e.g. $$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



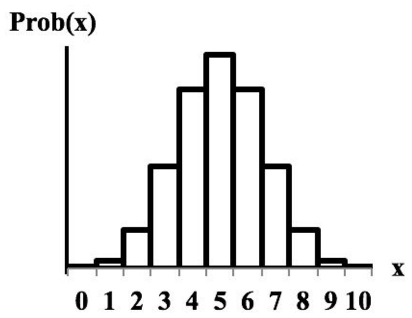
Discrete versus continuous probability

Discrete probability distribution

- Parameters: $p_1, p_2, \ldots, p_{n-1}$

- Probability mass:

$p(X=1)$	$p(X=2)$	$p(X=3)$
0.2	0.4	0.4

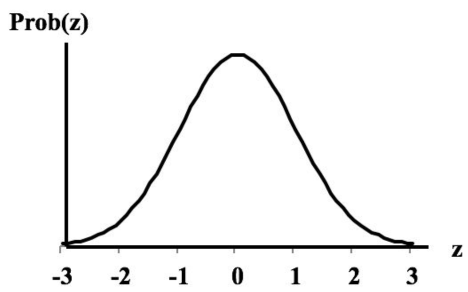


Continuous probability distribution

- Parameters: depends on distribution, e.g. normal distribution μ,σ

- Probability density:

e.g. $$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



Parameters define the entire distribution

2. Continuous Markov Decision Process

Sequential decision making

Sequential decision making

Discrete states

(board states)

Continuous states

(joint angles)

Sequential decision making

Discrete states

Continuous states

Discrete actions

(moves)

Continuous actions

(torques/voltages on motors)

Continuous Markov Decision Process

Principles stay largely the same

Continuous Markov Decision Process

Definition of MPD:

$$\{\mathcal{S}, \mathcal{A}, T(s'|s,a), R(s,a,s'), \gamma, p_0(s_0)\}$$

Continuous Markov Decision Process

Definition of MPD:

$$\{\mathcal{S}, \mathcal{A}, T(s'|s,a), R(s,a,s'), \gamma, p_0(s_0)\}$$

State space

Action space

Transition function

Reward function

Discount

Start state distribution

Continuous Markov Decision Process

Definition of MPD:

$$\{\mathcal{S}, \mathcal{A}, T(s'|s,a), R(s,a,s'), \gamma, p_0(s_0)\}$$

State space

Action space

Transition function

Reward function

Discount

Start state distribution

continuous spaces

continuous density
(if state space continuous)

Continuous Markov Decision Process

We define a policy:

Continuous Markov Decision Process

We define a policy:

- Stochastic policy $\pi(a|s)$ $\pi : \mathcal{S} \rightarrow p(\mathcal{A})$

Continuous Markov Decision Process

We define a policy:

- Stochastic policy $\pi(a|s)$ $\pi : \mathcal{S} \rightarrow p(\mathcal{A})$
- Deterministic policy $\pi(s)$ $\pi : \mathcal{S} \rightarrow \mathcal{A}$

Continuous Markov Decision Process

We define a policy:

- Stochastic policy $\pi(a|s)$ $\pi : \mathcal{S} \rightarrow p(\mathcal{A})$
- Deterministic policy $\pi(s)$ $\pi : \mathcal{S} \rightarrow \mathcal{A}$

If action space continuous: continuous density continuous number

Continuous Markov Decision Process

We define a policy:

- Stochastic policy $\pi(a|s)$ $\pi : \mathcal{S} \rightarrow p(\mathcal{A})$
- Deterministic policy $\pi(s)$ $\pi : \mathcal{S} \rightarrow \mathcal{A}$

If action space continuous: continuous density continuous number

Extensively discuss policy representation in next section

Continuous Markov Decision Process

We sample traces

$$h_t^n = \{s_t, a_t, r_t, s_{t+1}, .., a_{t+n}, r_{t+n}, s_{t+n+1}\}$$

Continuous Markov Decision Process

We sample traces

$$h_t^n = \{s_t, a_t, r_t, s_{t+1}, .., a_{t+n}, r_{t+n}, s_{t+n+1}\}$$

The cumulative return of the trace is

$$R(h_t) = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$$

Continuous Markov Decision Process

The average cumulative return are the **value** V(s) and **state-action value** Q(s,a)

Continuous Markov Decision Process

The average cumulative return are the **value** V(s) and **state-action value** Q(s,a)

$$V^{\pi}(s) = \mathbb{E}_{h_t \sim p(h_t)}\left[\sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} | s_t = s\right]$$

Continuous Markov Decision Process

The average cumulative return are the **value** V(s) and **state-action value** Q(s,a)

$$V^\pi(s) = \mathbb{E}_{h_t \sim p(h_t)}\Big[\sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} | s_t = s\Big]$$

$$Q^\pi(s,a) = \mathbb{E}_{h_t \sim p(h_t)}\Big[\sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} | s_t = s, a_t = a\Big]$$

Continuous Markov Decision Process

The average cumulative return are the **value** V(s) and **state-action value** Q(s,a)

$$V^\pi(s) = \mathbb{E}_{h_t \sim p(h_t)}\Big[\sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} | s_t = s\Big]$$

$$Q^\pi(s,a) = \mathbb{E}_{h_t \sim p(h_t)}\Big[\sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} | s_t = s, a_t = a\Big]$$

These value functions have recursive relations = Bellman equation

Continuous Bellman equations

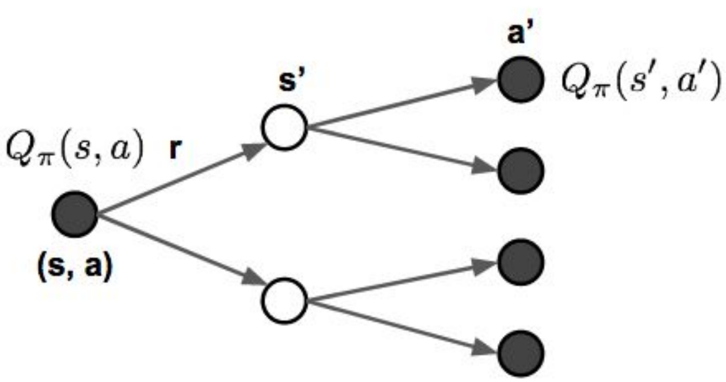
Bellman equation:

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\left[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\right]$$

Continuous Bellman equations

Bellman equation:

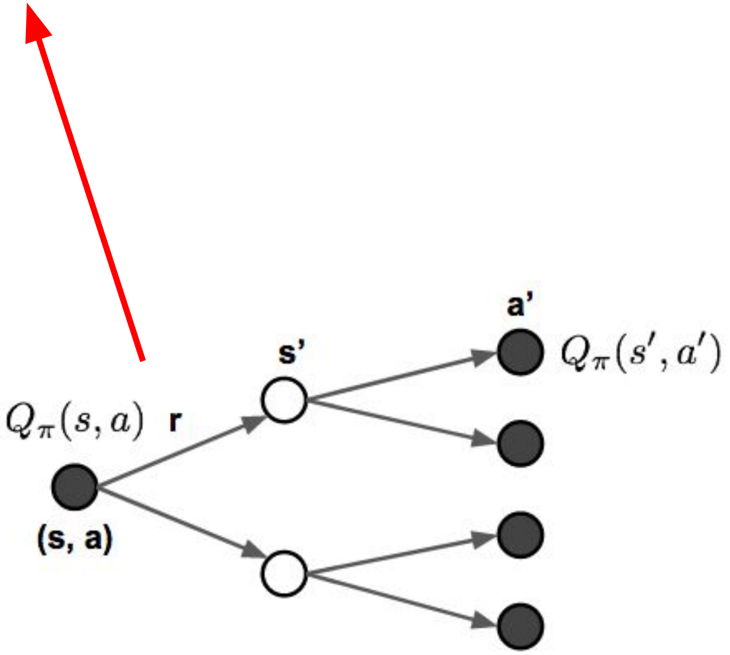
$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\left[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\right]$$



Continuous Bellman equations

Bellman equation:

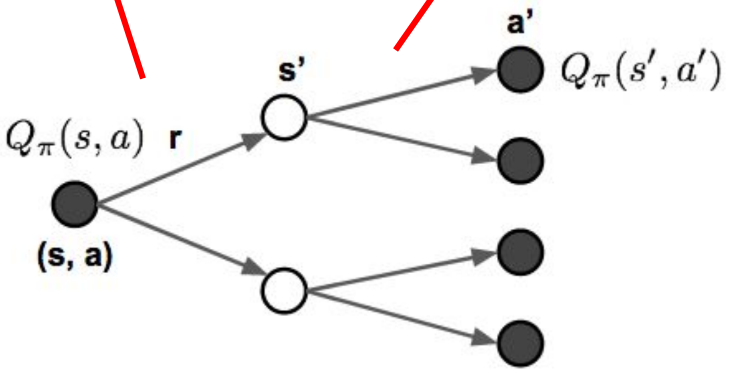
$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\left[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\right]$$



Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\left[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\right]$$



Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\left[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\right]$$

Discrete state & action space

$$Q(s,a) = \sum_{s' \in \mathcal{S}} T(s'|s,a)\left[r(s,a,s') + \gamma \cdot \sum_{a \in \mathcal{A}} \pi(a|s)[Q(s',a')]\right]$$

Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \boxed{\mathbb{E}_{s' \sim T(\cdot|a,s)}}\left[r(s,a,s') + \gamma \cdot \boxed{\mathbb{E}_{a' \sim \pi(\cdot|s')}}\left[Q(s',a')\right]\right]$$

Discrete state & action space

$$Q(s,a) = \boxed{\sum_{s' \in \mathcal{S}} T(s'|s,a)}\left[r(s,a,s') + \gamma \cdot \boxed{\sum_{a \in \mathcal{A}} \pi(a|s)}\left[Q(s',a')\right]\right]$$

Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\left[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\right]$$

Continuous state & action space

Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|a,s)}\big[r(s,a,s') + \gamma \cdot \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]\big]$$

Continuous state & action space

$$Q(s,a) = \int_{s'} T(s'|s,a)\Big[r(s,a,s') + \gamma \cdot \int_{a'} [\pi(a'|s') \cdot Q(s',a')]\, \mathrm{d}a'\Big]\, \mathrm{d}s'$$

Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \boxed{\mathbb{E}_{s' \sim T(\cdot|a,s)}}\left[r(s,a,s') + \gamma \cdot \boxed{\mathbb{E}_{a' \sim \pi(\cdot|s')}}[Q(s',a')]\right]$$

Continuous state & action space

$$Q(s,a) = \boxed{\int_{s'} T(s'|s,a)}\left[r(s,a,s') + \gamma \cdot \boxed{\int_{a'} [\pi(a'|s')}\cdot Q(s',a')]\,\mathrm{d}a'\right]\mathrm{d}s'$$

Continuous Bellman equations

Bellman equation:

$$Q(s,a) = \boxed{\mathbb{E}_{s'\sim T(\cdot|a,s)}}\left[r(s,a,s') + \gamma \cdot \boxed{\mathbb{E}_{a'\sim\pi(\cdot|s')}}\left[Q(s',a')\right]\right]$$

Continuous state & action space

$$Q(s,a) = \boxed{\int_{s'} T(s'|s,a)}\left[r(s,a,s') + \gamma \cdot \boxed{\int_{a'} [\pi(a'|s')}\cdot Q(s',a')]\,\mathrm{d}a'\right]\mathrm{d}s'$$

In expectations: summation becomes integration

Goal of RL & MDP search

Objective: Value of the start state (= expected cumulative reward)

Goal of RL & MDP search

Objective: Value of the start state (= expected cumulative reward)

$$J(\pi) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p(h_0|\pi)}\Big[R(h_0)\Big]$$

Goal of RL & MDP search

Objective: Value of the start state (= expected cumulative reward)

$$J(\pi) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p(h_0|\pi)}\Big[R(h_0)\Big]$$

function of policy (parameters)

Goal of RL & MDP search

Objective: Value of the start state (= expected cumulative reward)

$$J(\pi) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p(h_0|\pi)}\Big[R(h_0)\Big]$$

function of policy (parameters)

Goal: Find the optimal policy

Goal of RL & MDP search

Objective: Value of the start state (= expected cumulative reward)

$$J(\pi) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p(h_0|\pi)}\Big[R(h_0)\Big]$$

function of policy (parameters)

Goal: Find the optimal policy

$$\pi^{\star}(a|s) = \arg\max_{\pi} V^{\pi}(s_0)$$

Goal of RL & MDP search

Objective: Value of the start state (= expected cumulative reward)

$$J(\pi) = V^\pi(s_0) = \mathbb{E}_{h_0 \sim p(h_0|\pi)}\Big[R(h_0)\Big]$$

function of policy (parameters)

Goal: Find the optimal policy

$$\pi^\star(a|s) = \arg\max_\pi V^\pi(s_0)$$

policy that achieves the highest average cumulative reward

3. Representing the solution

Representing the solution

Policy = mapping from state to (probability distribution over) actions

Representing the solution

Policy = mapping from state to (probability distribution over) actions

=

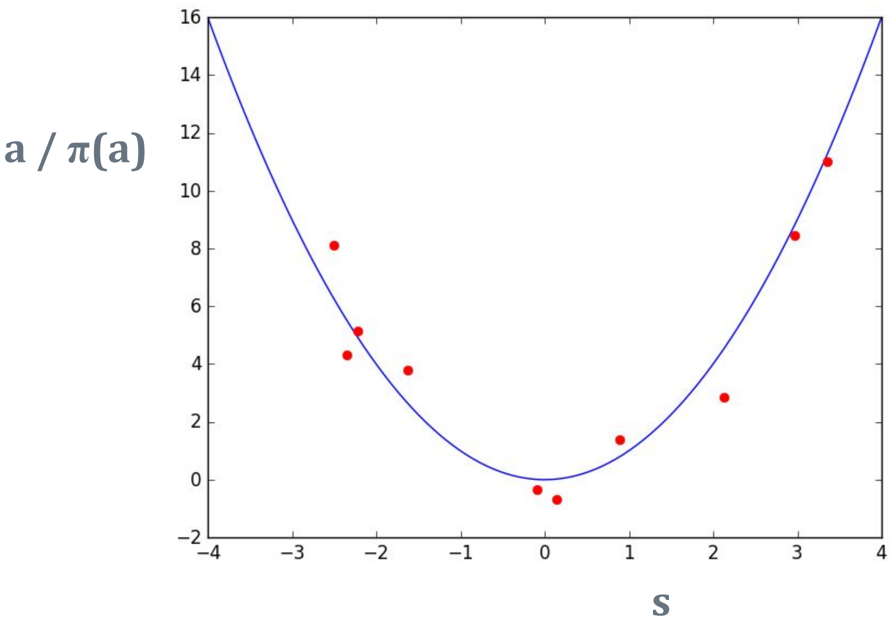
supervised learning problem

Representing the solution

Policy = mapping from state to (probability distribution over) actions

=

supervised learning problem



Supervised learning

Parametric
(e.g. table, neural network)

Non-parametric
(e.g. Gaussian Process, k-NN)

Supervised learning

$\boxed{\textbf{Parametric}}$

Non-parametric

$$\pi_\theta(a|s)$$

$$\pi_\theta : \mathcal{S} \times \Theta \to p(\mathcal{A}).$$

Parametric supervised learning

Parametric supervised learning

Parametric model type	**Tabular**	**Function approximation**

Parametric supervised learning

Parametric model type	**Tabular** e.g., table, parameters are entries (*planning & RL*)	**Function approximation** e.g. neural network (*RL*)

Parametric supervised learning

Parametric model type	**Tabular** e.g., table, parameters are entries (*planning & RL*)	**Function approximation** e.g. neural network (*RL*)
Validity	**Global**	**Local**

Parametric supervised learning

Parametric model type	**Tabular** e.g., table, parameters are entries (*planning & RL*)	**Function approximation** e.g. neural network (*RL*)
Validity	**Global** solution for entire input space (*RL*)	**Local** solution for local region of input space (*planning*)

Parametric supervised learning

Parametric model type	**Tabular** e.g., table, parameters are entries (*planning & RL*)	**Function approximation** e.g. neural network (*RL*)
Validity	**Global** solution for entire input space (*RL*)	**Local** solution for local region of input space (*planning*)

Parametric supervised learning

Parametric model type	**Tabular** e.g., table, parameters are entries (*planning & RL*)	**Function approximation** e.g. neural network (*RL*)
	?	
Validity	**Global** solution for entire input space (*RL*)	**Local** solution for local region of input space (*planning*)

Representing the solution

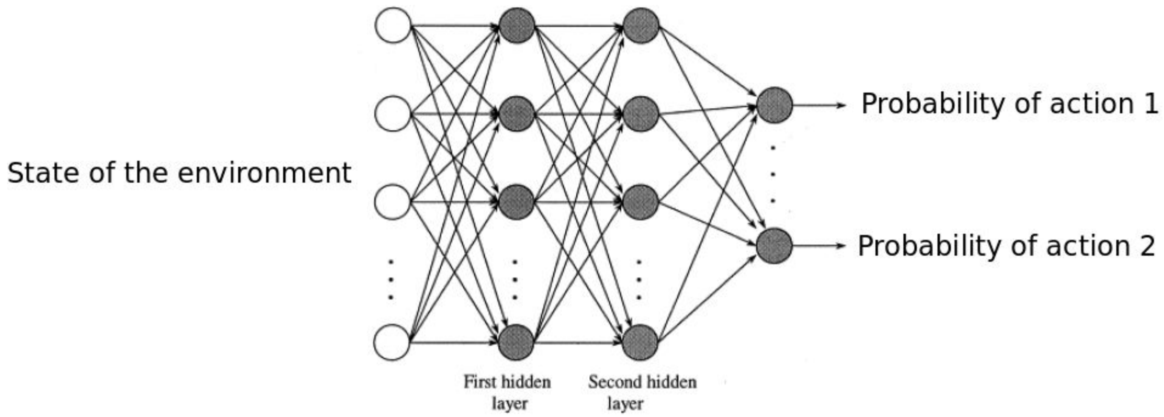
<u>How should we represent this function?</u>

$$\pi : \mathcal{S} \to p(\mathcal{A})$$

Representing the solution

<u>How should we represent this function?</u>

$$\pi : \mathcal{S} \rightarrow p(\mathcal{A})$$

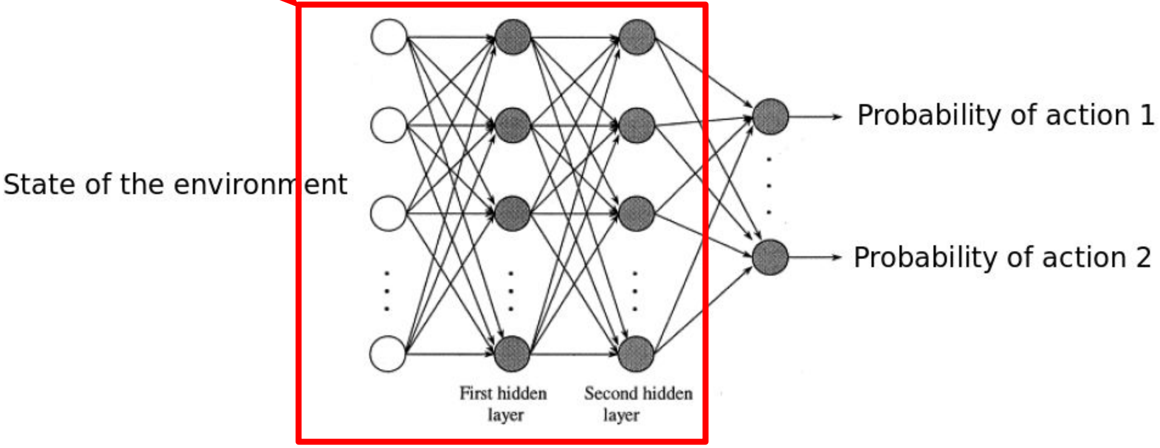


Representing the solution

How should we represent this function?

$$\pi \boxed{: \mathcal{S}} \to p(\mathcal{A})$$

Input space characteristics mostly determine model type



Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

1. Type of input space

2. Dim of input space		Discrete	Continuous
	Low dimensional		
	High dimensional		

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

1. Type of input space

2. Dim of input space

	Discrete	Continuous
Low dimensional		
High dimensional		

Tabular representation

Tabular representation

Policy table

s	$\pi(a\text{=up}\mid s)$	$\pi(a\text{=down}\mid s)$	$\pi(a\text{=left}\mid s)$	$\pi(a\text{=right}\mid s)$
1	0.2	0.8	0.0	0.0
2	0.0	0.0	0.0	1.0
3	0.7	0.0	0.3	0.0
etc.	..	..	..	..

Tabular representation

Policy table

s	$\pi(a\text{=up}\|s)$	$\pi(a\text{=down}\|s)$	$\pi(a\text{=left}\|s)$	$\pi(a\text{=right}\|s)$
1	0.2	0.8	0.0	0.0
2	0.0	0.0	0.0	1.0
3	0.7	0.0	0.3	0.0
etc.	..	..	..	..

State-action value table

	a=up	a=down	a=left	a=right
s=1	4.0	3.0	7.0	1.0
s=2	2.0	-4.0	0.3	1.0
s=3	3.5	0.8	3.6	6.2
etc.	..	..	..	..

Tabular representation

Policy table

s	$\pi(a{=}\text{up}\|s)$	$\pi(a{=}\text{down}\|s)$	$\pi(a{=}\text{left}\|s)$	$\pi(a{=}\text{right}\|s)$
1	0.2	0.8	0.0	0.0
2	0.0	0.0	0.0	1.0
3	0.7	0.0	0.3	0.0
etc.	..	..	..	..

State-action value table

	a=up	a=down	a=left	a=right
s=1	4.0	3.0	7.0	1.0
s=2	2.0	-4.0	0.3	1.0
s=3	3.5	0.8	3.6	6.2
etc.	..	..	..	..

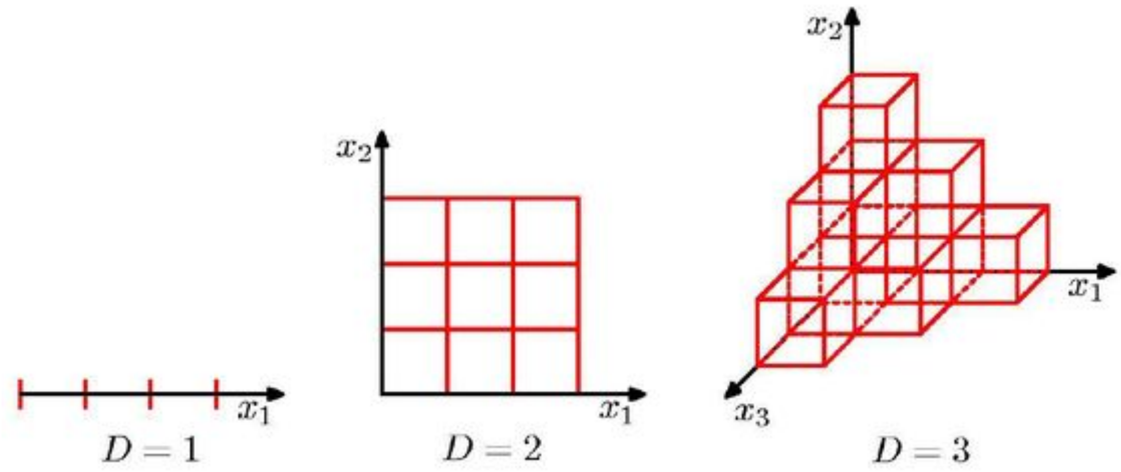
Table entries are the parameters!

Curse of dimensionality

The size/cardinality of a space scales exponentially in it dimensionality

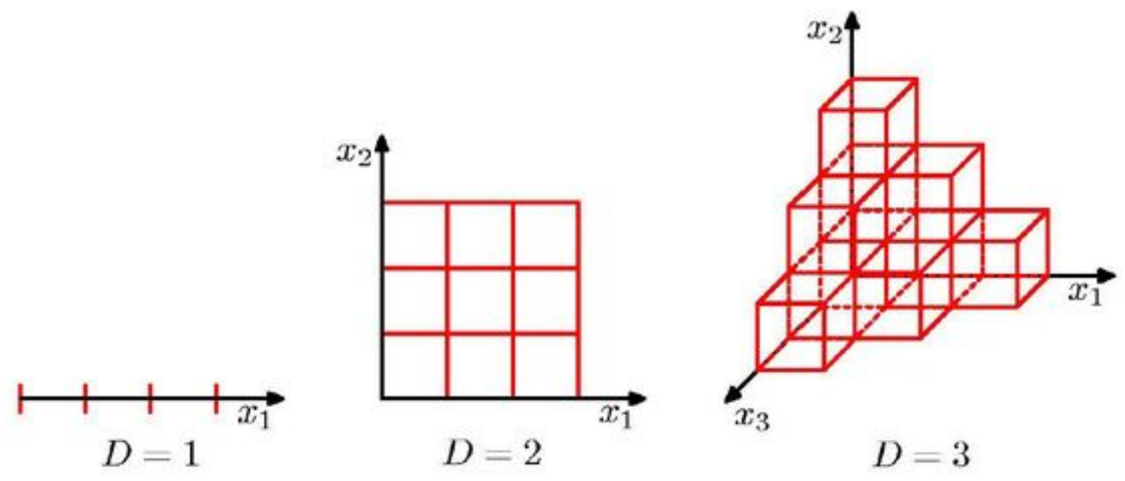
Curse of dimensionality

The size/cardinality of a space scales exponentially in it dimensionality



Curse of dimensionality

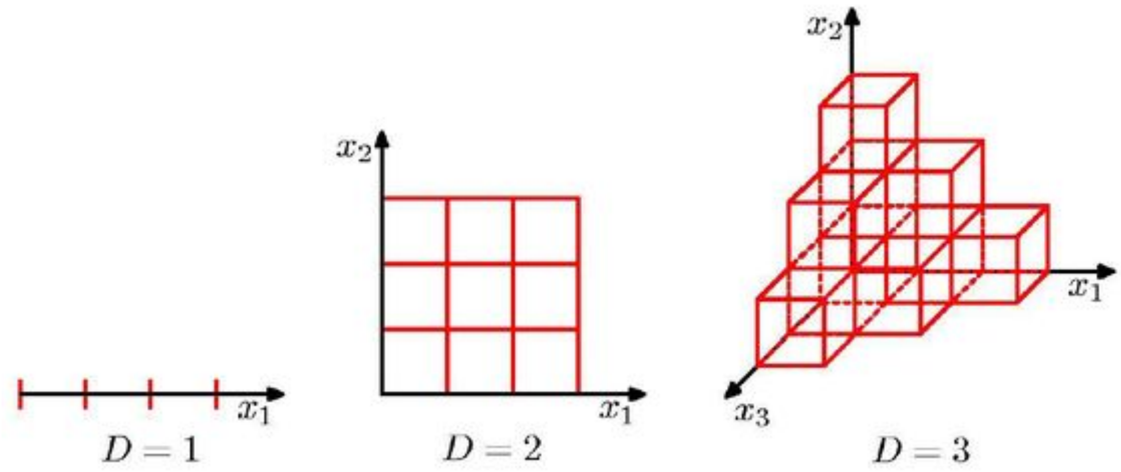
The size/cardinality of a space scales exponentially in it dimensionality



$$|\mathcal{X}| = O\big(\exp(\mathrm{Dim}(\mathcal{X}))\big)$$

Curse of dimensionality

The size/cardinality of a space scales exponentially in it dimensionality



$$|\mathcal{X}| = O\big(\exp(\mathrm{Dim}(\mathcal{X}))\big)$$

Problem for tables!

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

1. Type of input space

2. Dim of input space		Discrete	Continuous
	Low dimensional	Table(/FA)	
	High dimensional		

Tabular or function approximation

Two main aspects of S:

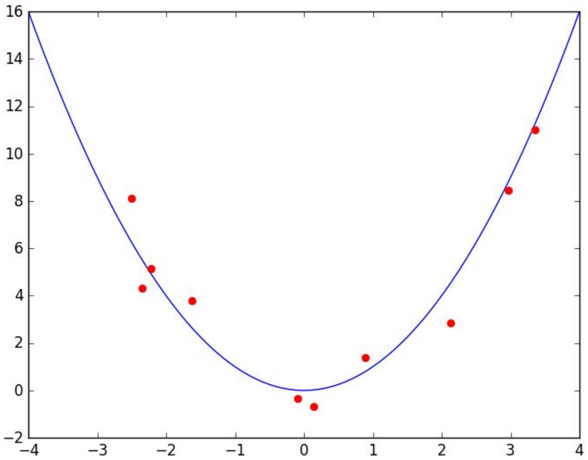
1. Type of space
2. Dimensionality

1. Type of input space

2. Dim of input space		Discrete	Continuous
	Low dimensional	Table(/FA)	
	High dimensional		

Function approximation

1. Specify parametrized function from input to output
2. Specify objective/loss
3. Optimize parameters to minimize loss



Examples: linear regression, neural networks

Tabular versus function approximation

	Benefit	Problem
Table		
Function approximation		

Tabular versus function approximation

	Benefit	Problem
Table	• Exact • (Easy to design)	
Function approximation		

Tabular versus function approximation

	Benefit	Problem
Table	• Exact • (Easy to design)	• Curse of dimensionality • No generalization
Function approximation		

Tabular versus function approximation

	Benefit	Problem
Table	• Exact • (Easy to design)	• Curse of dimensionality • No generalization
Function approximation	• Generalization • Lower memory requirement (scales to high-dim)	

Tabular versus function approximation

	Benefit	Problem
Table	• Exact • (Easy to design)	• Curse of dimensionality • No generalization
Function approximation	• Generalization • Lower memory requirement (scales to high-dim)	• Approximation errors

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

1. Type of input space

2. Dim of input space		Discrete	Continuous
	Low dimensional	Table(/FA)	
	High dimensional	$\boxed{\text{FA}}$	

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

1. Type of input space

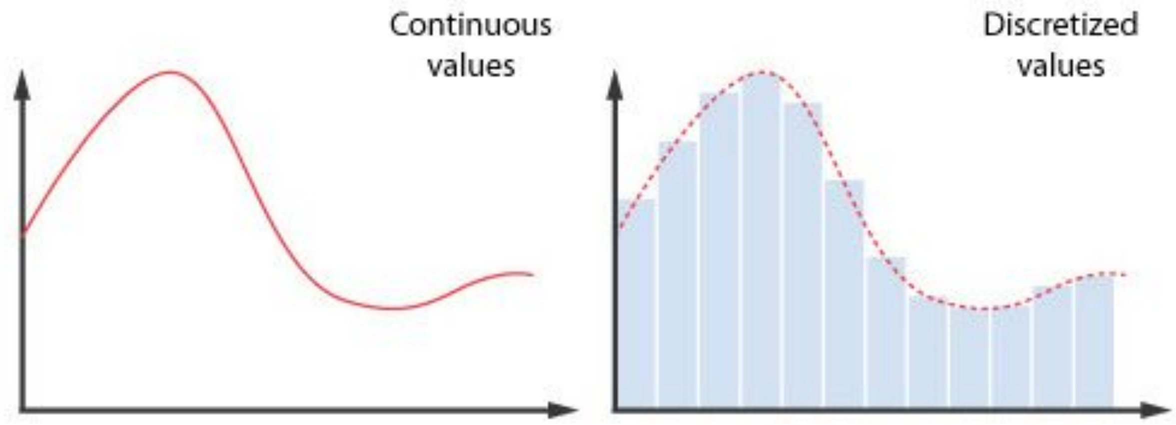
2. Dim of input space		Discrete	Continuous
	Low dimensional	Table(/FA)	
	High dimensional	FA	

Discretization

A first approach to store the solution for continuous input spaces is discretization

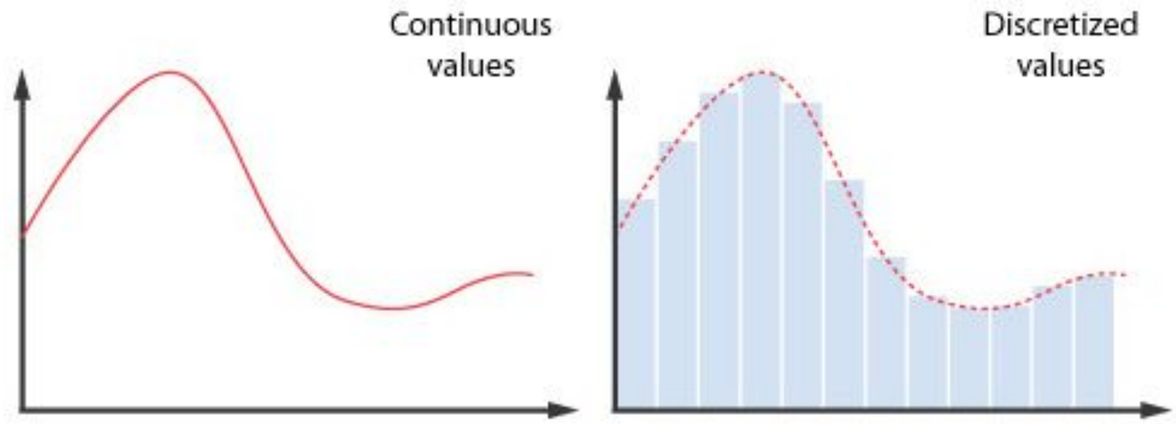
Discretization

A first approach to store the solution for continuous input spaces is discretization



Discretization

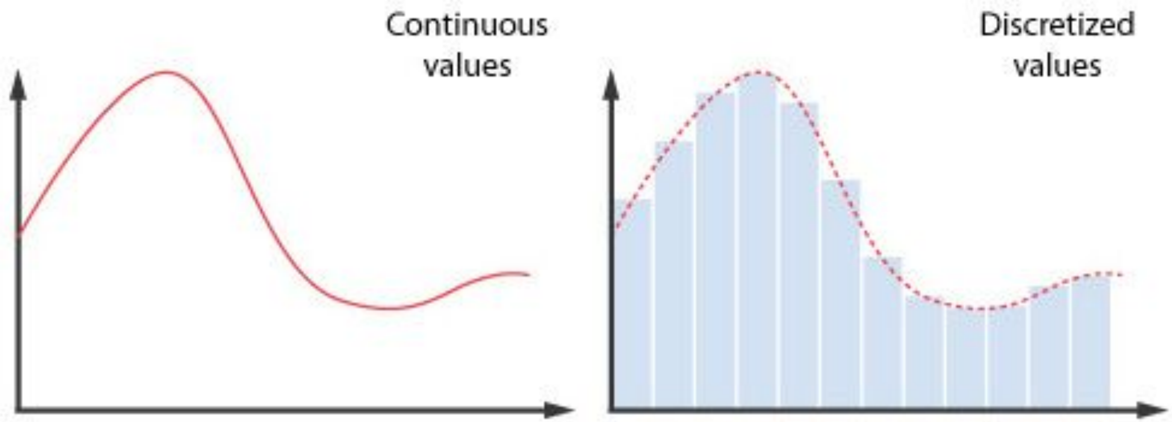
A first approach to store the solution for continuous input spaces is discretization



Works for low-dimensional input (state) spaces,
but also suffers from the curse of dimensionality

Discretization

A first approach to store the solution for continuous input spaces is discretization



As a second solution, function approximation always works for continuous input & scales to high dimensions

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

2. Dim of input space		1. Type of input space	
		Discrete	**Continuous**
	Low dimensional	Table(/FA)	Discretization/FA
	High dimensional	FA	FA

Tabular or function approximation

Two main aspects of S:

1. Type of space
2. Dimensionality

1. Type of input space

2. Dim of input space		Discrete	Continuous
	Low dimensional	Table(/FA)	Discretization/FA
	High dimensional	FA	FA

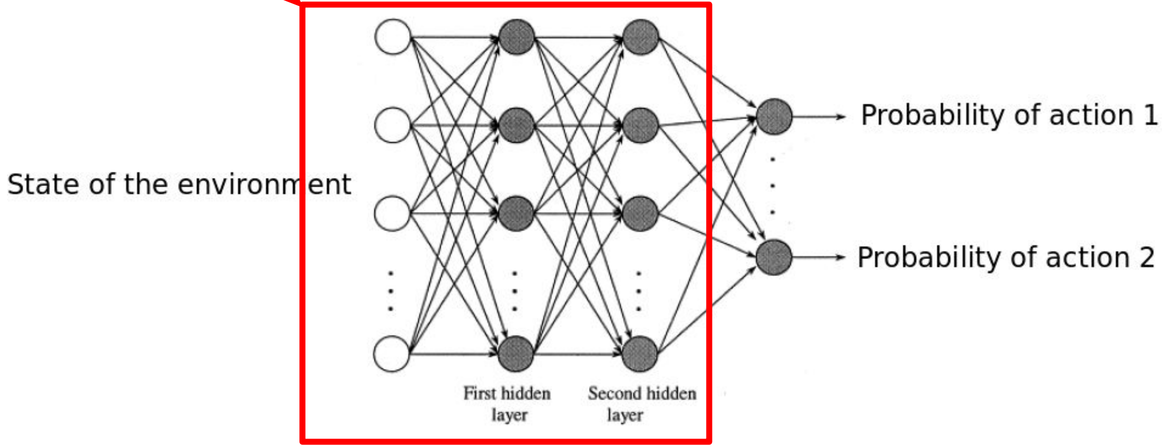
In short: small problems discretization, larger problems function approximation

Representing the solution

How should we represent this function?

$$\pi : \boxed{\mathcal{S}} \to p(\mathcal{A})$$

Input space characteristics mostly determine model type

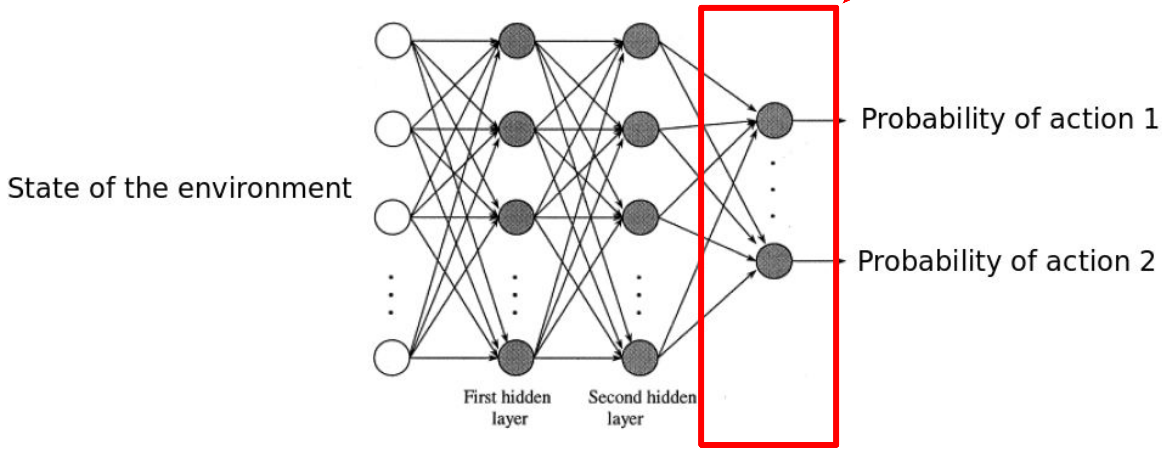


Representing the solution

How should we represent this function?

$$\pi : \mathcal{S} \rightarrow \boxed{p(\mathcal{A})}$$

Output space determines type of policy



Policy output considerations

Three main considerations for A:

Policy output considerations

Three main considerations for A:

1. Type of action space: discrete - continuous

Policy output considerations

Three main considerations for A:

1. Type of action space: discrete - continuous
2. Choice of representation: implicit (value-based) - explicit (policy-based) - both (actor-critic)

Policy output considerations

Three main considerations for A:

1.	Type of action space:	discrete - continuous
2.	Choice of representation:	implicit (value-based) - explicit (policy-based) - both (actor-critic)
3.	Choice of probabilistic:	stochastic - deterministic

Policy output considerations

Three main considerations for A:

1. Type of action space: discrete - continuous
2. Choice of representation: implicit (value-based) - explicit (policy-based) - both (actor-critic)
3. Choice of probabilistic: stochastic - deterministic

Dimensionality of A is usually low & not an important factor

Policy output considerations

Three main considerations for A:

1. Type of action space: discrete - continuous
2. Choice of representation: implicit (value-based) - explicit (policy-based) - both (actor-critic)
3. Choice of probabilistic: stochastic - deterministic

	Implicit policy $\pi = f(Q_\theta(s,a))$	**Explicit policy** $\pi_\theta(a\|s)$
Discrete action space		
Continuous action space		

Policy output considerations

Three main considerations for A:

1. Type of action space: discrete - continuous
2. Choice of representation: implicit (value-based) - explicit (policy-based) - both (actor-critic)
3. Choice of probabilistic: stochastic - deterministic

	Implicit policy $\pi = f(Q_\theta(s, a))$	**Explicit policy** $\pi_\theta(a\|s)$
Discrete action space		
Continuous action space		

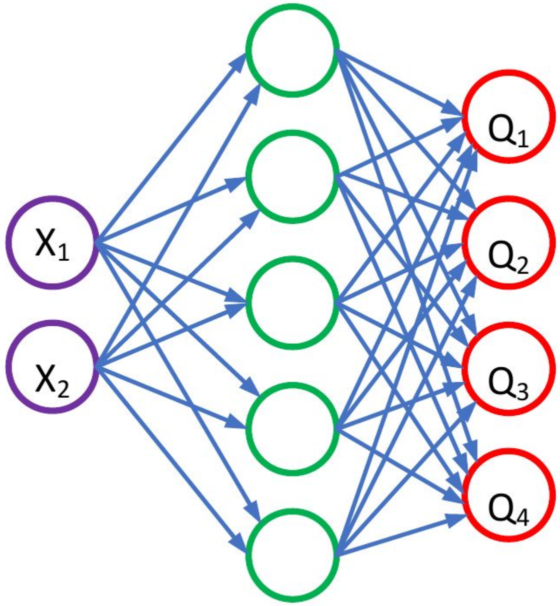
Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

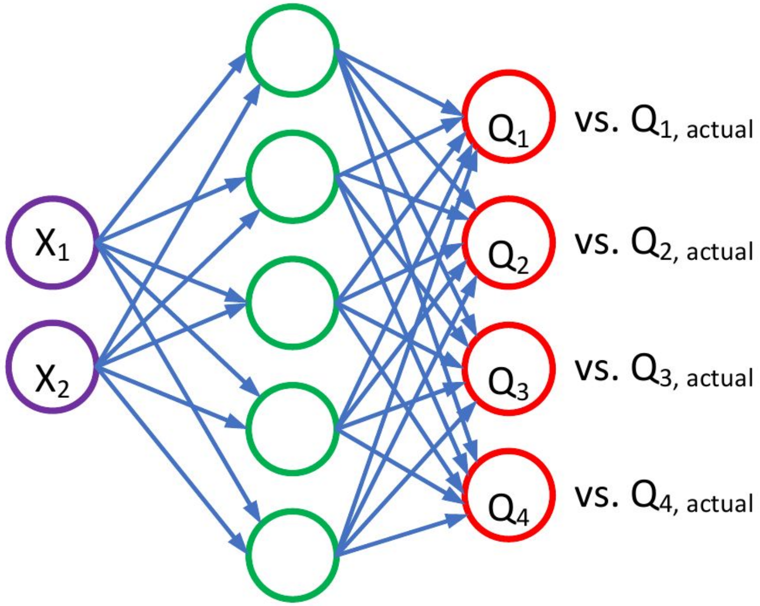
Learn a state(-action) value function



Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Learn a state(-action) value function



Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Learn a state(-action) value function

Make policy a hardcoded function of value function

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Deterministic examples

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Deterministic examples

$$\pi_\theta(s) = \arg\max_{a \in \mathcal{A}} Q_\theta(s,a)$$

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Deterministic examples

$$\pi_\theta(s) = \arg\max_{a \in \mathcal{A}} Q_\theta(s,a)$$

“greedy policy”

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Deterministic examples

$$\pi_\theta(s) = \arg\max_{a \in \mathcal{A}} Q_\theta(s,a)$$

"greedy policy"

$$\pi_\theta(s) = \arg\max_{a} \left[\bar{Q}_\theta(s,a) + c \cdot \sqrt{\frac{\ln n(s)}{n(s,a)}} \right]$$

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Deterministic examples

$$\pi_\theta(s) = \arg\max_{a \in \mathcal{A}} Q_\theta(s,a)$$

"greedy policy"

$$\pi_\theta(s) = \arg\max_a \left[\bar{Q}_\theta(s,a) + c \cdot \sqrt{\frac{\ln n(s)}{n(s,a)}} \right]$$

"UCT policy"
(MCTS)

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Deterministic examples

$$\pi_\theta(s) = \arg\max_{a \in \mathcal{A}} Q_\theta(s,a)$$

"greedy policy"

$$\pi_\theta(s) = \arg\max_a \left[\bar{Q}_\theta(s,a) + c \cdot \sqrt{\frac{\ln n(s)}{n(s,a)}} \right]$$

"UCT policy"
(MCTS)

Note that the UCT policy is stochastic over multiple samples, since n(s,a) will change over multiple steps!

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Stochastic example

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Stochastic example

$$\pi_\theta(a|s) = \frac{\exp Q_\theta(s,a)/\tau}{\sum_{b\in\mathcal{A}} \exp Q_\theta(s,b)/\tau}$$

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Stochastic example

$$\pi_\theta(a|s) = \frac{\exp Q_\theta(s,a)/\tau}{\sum_{b\in\mathcal{A}} \exp Q_\theta(s,b)/\tau}$$

"Boltzmann policy"

Implicit policies (= value-based RL)

$$\pi_\theta(a|s) = f(Q_\theta(s,a))$$

Stochastic example

$$\pi_\theta(a|s) = \frac{\exp Q_\theta(s,a)/\tau}{\sum_{b\in\mathcal{A}} \exp Q_\theta(s,b)/\tau}$$

"Boltzmann policy"

All examples for discrete action space!

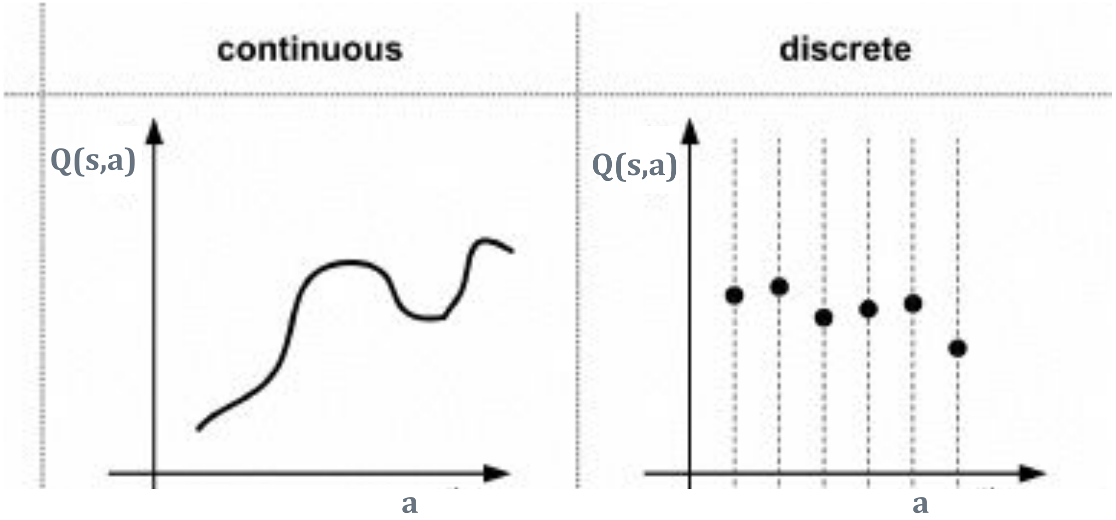
Does it also work for continuous action spaces?

Implicit policies (= value-based RL)

All implicit policies require a form of maximization
(to give better actions higher chance of selection)

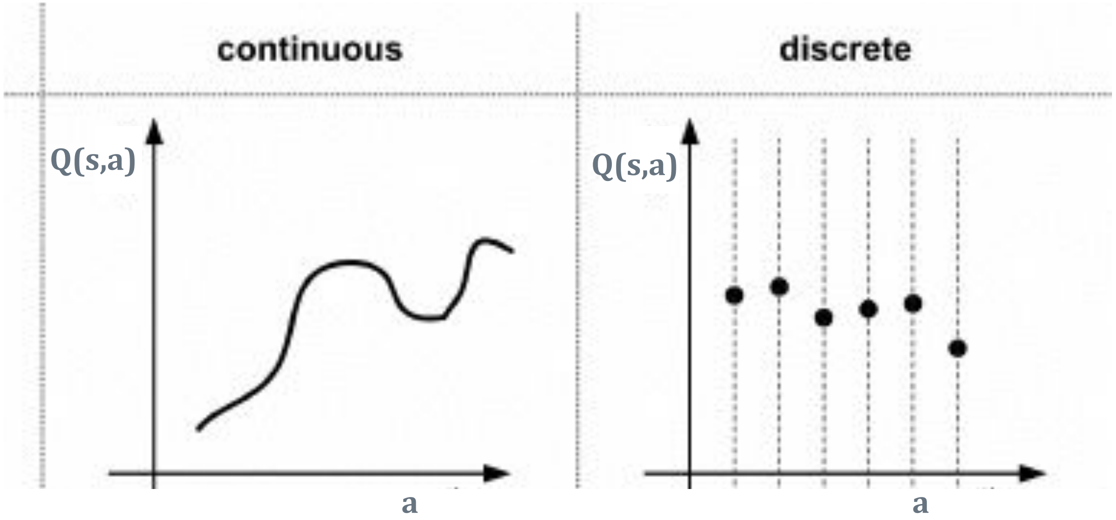
Implicit policies (= value-based RL)

All implicit policies require a form of maximization
(to give better actions higher chance of selection)



Implicit policies (= value-based RL)

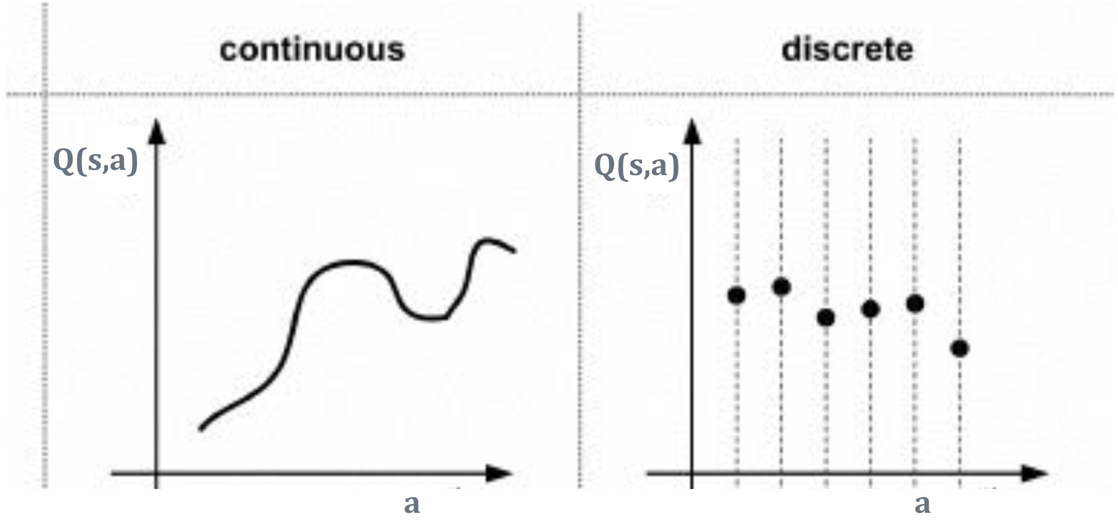
All implicit policies require a form of maximization
(to give better actions higher chance of selection)



Discrete action space:
easy maximization

Implicit policies (= value-based RL)

All implicit policies require a form of maximization
(to give better actions higher chance of selection)



Continuous action space:
tough! (entire new maximization)

Discrete action space:
easy maximization

Explicit versus implicit policy

Two main aspects of A:

1. Type of policy (implicit or explicit)
2. Type of space (discrete versus continuous)

	Implicit policy $\pi = f(Q_\theta(s, a))$	Explicit policy $\pi_\theta(a\|s)$
Discrete action space	✓	
Continuous action space	-	

Explicit versus implicit policy

Two main aspects of A:

1. Type of policy (implicit or explicit)
2. Type of space (discrete versus continuous)

	Implicit policy $\pi = f(Q_\theta(s, a))$	**Explicit policy** $\pi_\theta(a\|s)$
Discrete action space	✓	
Continuous action space	$\boxed{-}$	

Implicit policies (as widely used in search and value-based RL)
not really useful in continuous action space!

Explicit versus implicit policy

Two main aspects of A:

1. Type of policy (implicit or explicit)
2. Type of space (discrete versus continuous)

	Implicit policy $\pi = f(Q_\theta(s, a))$	**Explicit policy** $\pi_\theta(a\|s)$
Discrete action space	✓	
Continuous action space	-	

Explicit policies (= policy-based RL)

$$Direct\ map\ S\ to\ p(A)\ or\ A$$

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Discrete action space:

- *Deterministic discrete policy*

  = *uncommon, since argmax is not differentiable*

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Discrete action space:

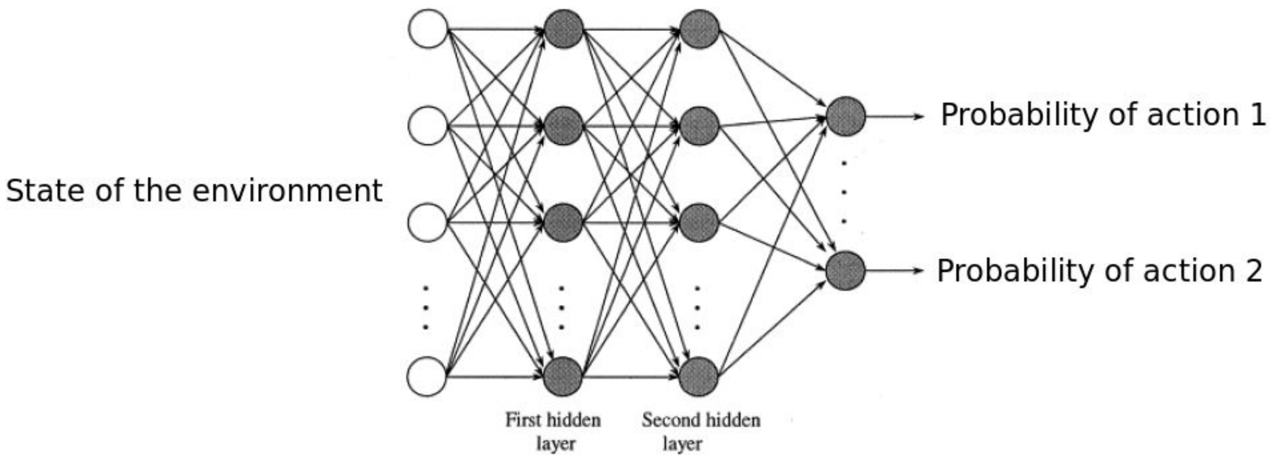
- *Stochastic,* **e.g.** $\pi_\theta(a|s) = \mathrm{softmax}(f_\theta(s))$

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Discrete action space:

e.g. $\pi_\theta(a|s) = \mathrm{softmax}(f_\theta(s))$

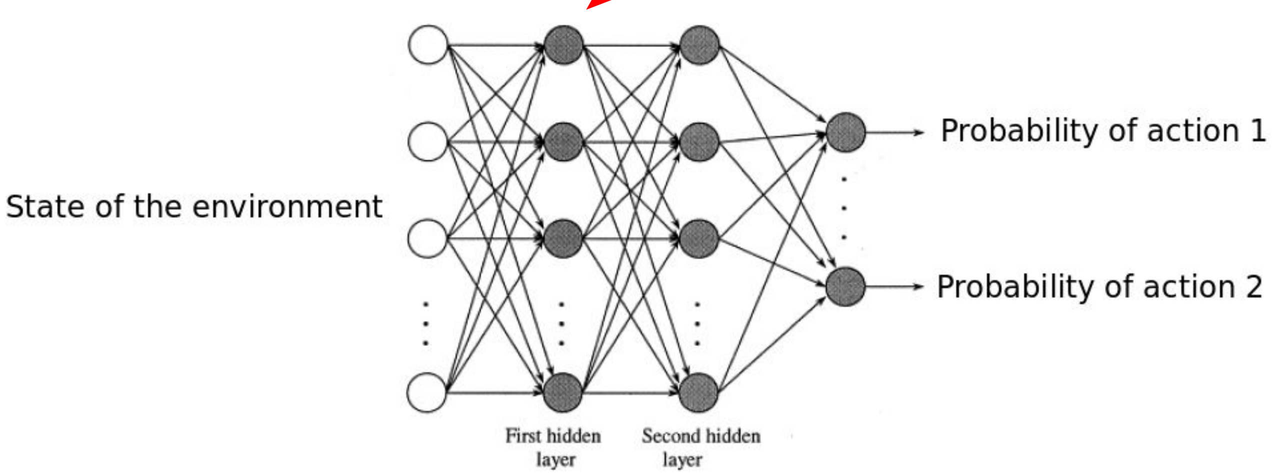


Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Discrete action space:

e.g. $\pi_\theta(a|s) = \mathrm{softmax}(f_\theta(s))$



Explicit policies (= policy-based RL)

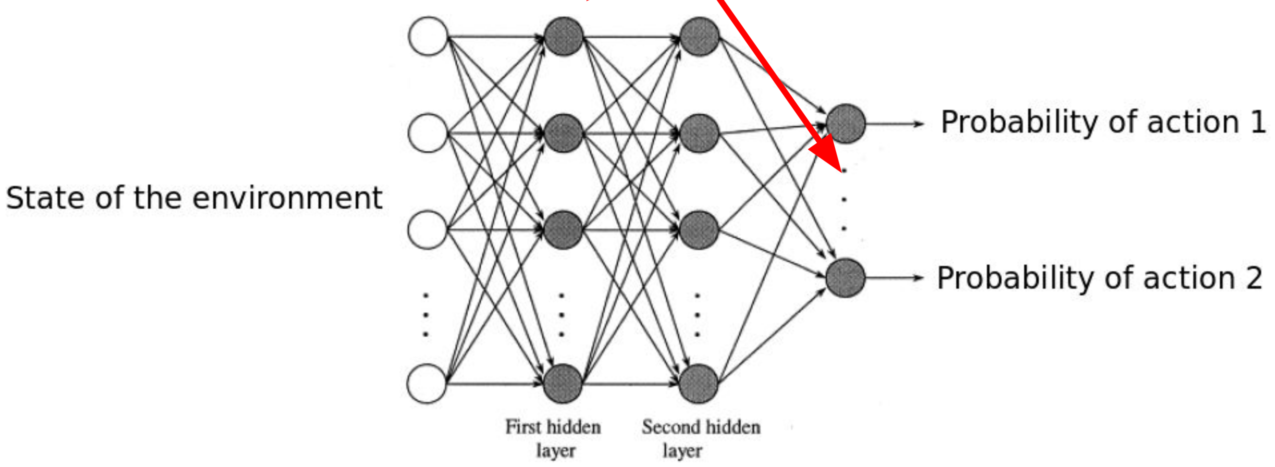
Direct map S to p(A) or A

$$\text{softmax}(y) = \frac{e^{y_i}}{\sum_k e^{y_k}}.$$

Discrete action space:

Softmax turns vector of real numbers into discrete probability distribution

e.g. $\pi_\theta(a|s) = \text{softmax}(f_\theta(s))$



Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

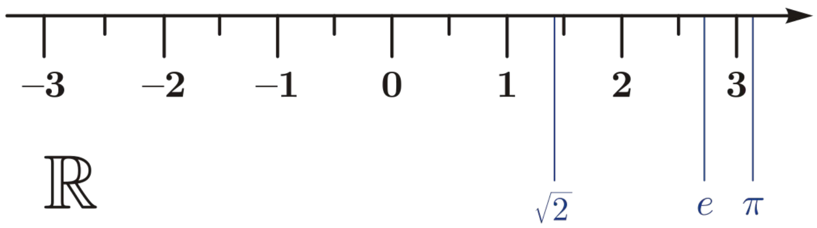
- Deterministic policy: simply predict a real number

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Deterministic policy: simply predict a real number



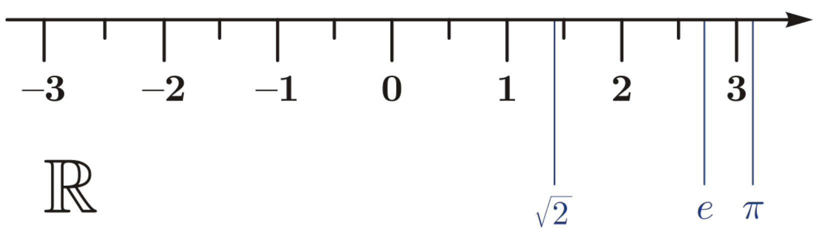
Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Deterministic policy: simply predict a real number

To make it a probability: can think of it as the mean of a Gaussian with fixed variance

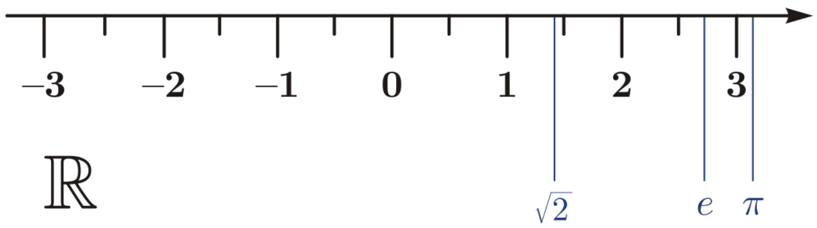


Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Deterministic policy: simply predict a real number



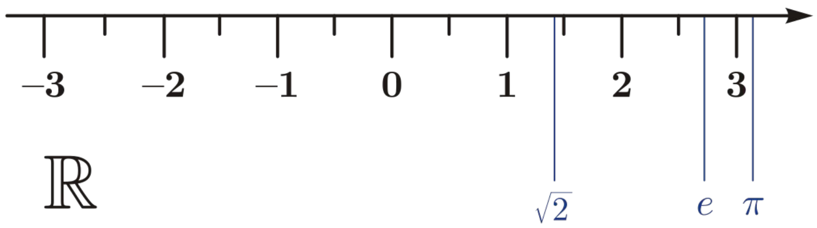
Important: continuous action spaces are usually *bounded.*

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Deterministic policy: simply predict a real number



Important: continuous action spaces are usually *bounded.*

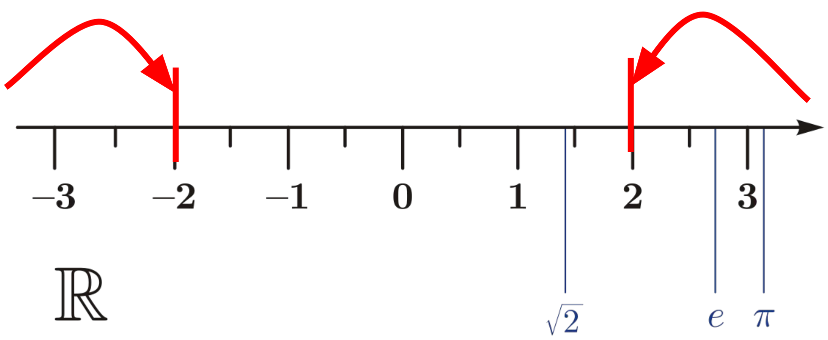
Solution: clipping (when prediction falls outside range, simply act as if it was at the boundary)

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Deterministic policy: simply predict a real number



Important: continuous action spaces are usually *bounded.*

Solution: clipping (when prediction falls outside range, simply act as if it was at the boundary)

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

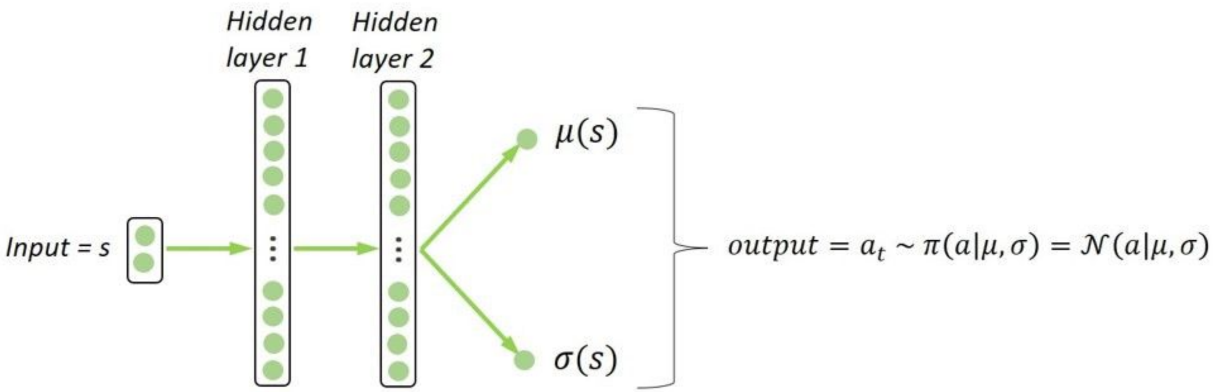
- Stochastic policy: predict the parameters of a continuous distribution!

Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Stochastic policy: predict the parameters of a continuous distribution!

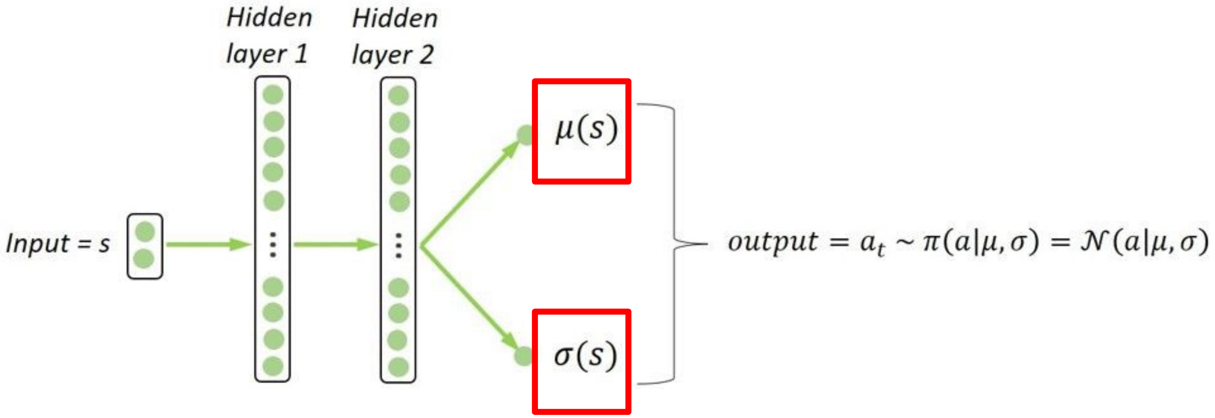


Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Stochastic policy: predict the parameters of a continuous distribution!



Explicit policies (= policy-based RL)

Direct map S to p(A) or A

Continuous action space

- Stochastic policy: predict the parameters of a continuous distribution!

$$\mu_\theta(s) = f_\theta(s)$$

$$\sigma_\theta(s) = \exp f_\theta(s) \text{ or } \sigma_\theta(s) = \text{softplus}(f_\theta(s))$$

Explicit policies (= policy-based RL)

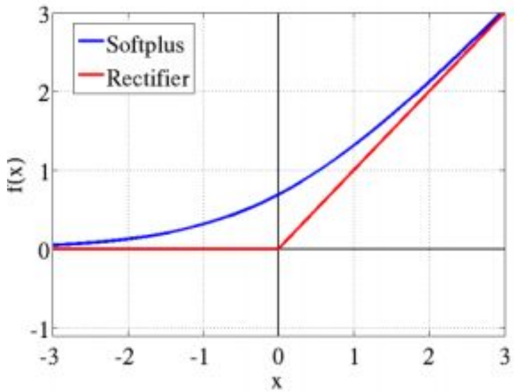
Direct map S to p(A) or A

Continuous action space

- Stochastic policy: predict the parameters of a continuous distribution!

$$\mu_\theta(s) = f_\theta(s)$$

$$\sigma_\theta(s) = \exp f_\theta(s) \text{ or } \sigma_\theta(s) = \text{softplus}(f_\theta(s))$$



Some parameters have restrictions, e.g., a standard deviation should be positive

Actor-critic (= value + policy)

Can combine an explicit policy and a value network!

Actor-critic (= value + policy)

Can combine an explicit policy and a value network!

Usually the value function will help the policy update:

1. For bootstrapping
2. For baseline subtraction
3. As a direct target to maximize

Actor-critic (= value + policy)

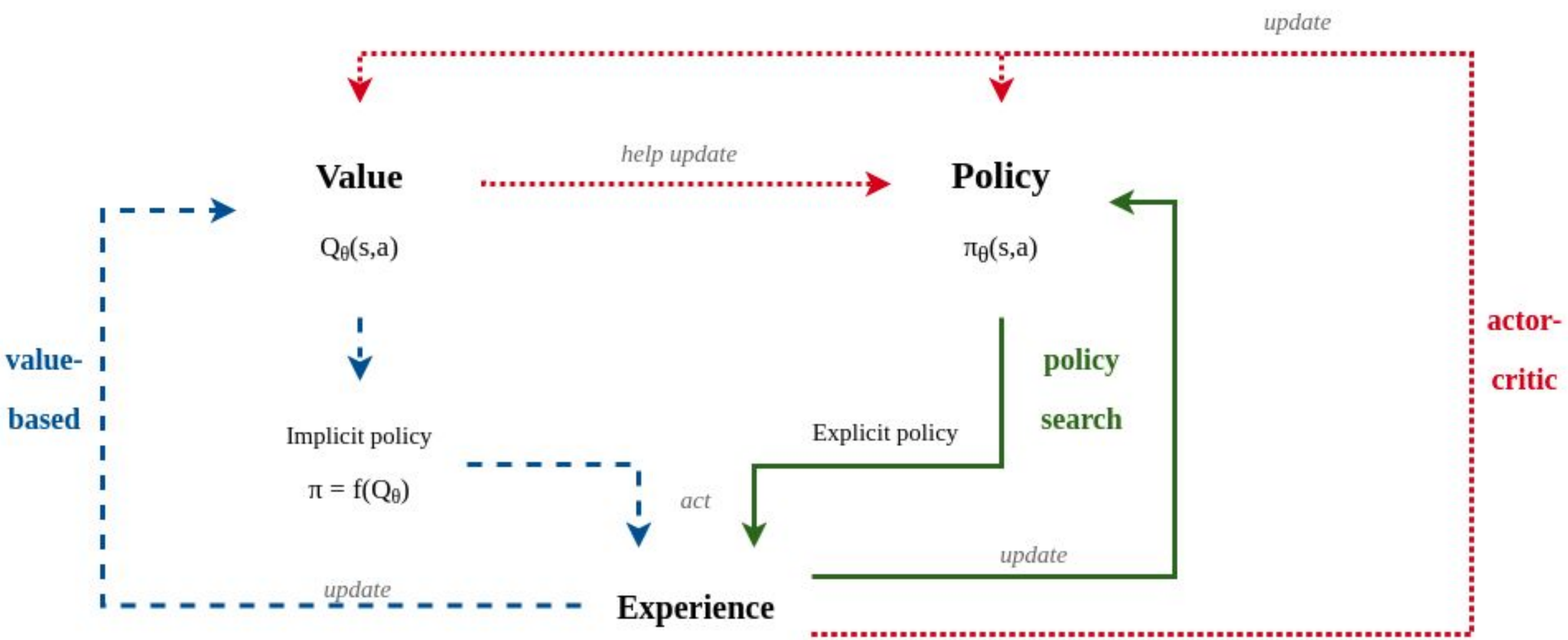
Can combine an explicit policy and a value network!

Usually the value function will help the policy update:

1. For bootstrapping
2. For baseline subtraction
3. As a direct target to maximize

We will encounter these settings after the break

Three representations of solution



Three representations of solution

		Discrete action space	Continuous action space
Value-based	**Stochastic**	e.g., $\pi_\theta(a\|s) = \epsilon\text{-greedy}(Q_\theta(s,a))$	$(\text{-})^\circ$
	Deterministic	e.g., $\pi_\theta(s) = \arg\max_a Q_\theta(s,a)$	$(\text{-})^\circ$
Policy search	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$
	Deterministic	$(\text{-})^\triangle$	e.g., $\pi_\theta(s) = f_\theta(s)$
Actor-critic	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$ & $V_\theta(s) = f_\theta(s)$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$ & $V_\theta(s) = f_\theta(s)$
	Deterministic	$(\text{-})^\triangle$	e.g., $\pi_\theta(s) == f_\theta(s)$ & $V_\theta(s) = g_\theta(s)$

Three representations of solution

Property of environment

		Discrete action space	Continuous action space
Value-based	**Stochastic**	e.g., $\pi_\theta(a\|s) = \epsilon\text{-greedy}(Q_\theta(s,a))$	(-)°
	Deterministic	e.g., $\pi_\theta(s) = \arg\max_a Q_\theta(s,a)$	(-)°
Policy search	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$
	Deterministic	(-)$^\triangle$	e.g., $\pi_\theta(s) = f_\theta(s)$
Actor-critic	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$ & $V_\theta(s) = f_\theta(s)$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$ & $V_\theta(s) = f_\theta(s)$
	Deterministic	(-)$^\triangle$	e.g., $\pi_\theta(s) == f_\theta(s)$ & $V_\theta(s) = g_\theta(s)$

Three representations of solution

Property of environment

Own choices

		Discrete action space	Continuous action space
Value-based	**Stochastic**	e.g., $\pi_\theta(a\|s) = \epsilon\text{-greedy}(Q_\theta(s,a))$	(-)°
	Deterministic	e.g., $\pi_\theta(s) = \arg\max_a Q_\theta(s,a)$	(-)°
Policy search	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$
	Deterministic	(-)$^\triangle$	e.g., $\pi_\theta(s) = f_\theta(s)$
Actor-critic	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$ & $V_\theta(s) = f_\theta(s)$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$ & $V_\theta(s) = f_\theta(s)$
	Deterministic	(-)$^\triangle$	e.g., $\pi_\theta(s) == f_\theta(s)$ & $V_\theta(s) = g_\theta(s)$

Three representations of solution

Property of environment

Own choices

		Discrete action space	Continuous action space
Value-based	**Stochastic**	e.g., $\pi_\theta(a\|s) = \epsilon\text{-greedy}(Q_\theta(s,a))$	(-)°
	Deterministic	e.g., $\pi_\theta(s) = \arg\max_a Q_\theta(s,a)$	(-)°
Policy search	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$
	Deterministic	(-)△	e.g., $\pi_\theta(s) = f_\theta(s)$
Actor-critic	**Stochastic**	e.g. $\pi_\theta(a\|s) = \text{softmax}(f_\theta(s,a))$ & $V_\theta(s) = f_\theta(s)$	e.g., $\pi_\theta(a\|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$ & $V_\theta(s) = f_\theta(s)$
	Deterministic	(-)△	e.g., $\pi_\theta(s) == f_\theta(s)$ & $V_\theta(s) = g_\theta(s)$

No need to understand this now, full summary in lecture notes

Summary first part

1. **Continuous state space:**

2. **Continuous action space**:

Summary first part

1. **Continuous state space:**

   Small = discretization, large = function approximation

2. **Continuous action space**:

Summary first part

1. **Continuous state space:**

   Small = discretization, large = function approximation

2. **Continuous action space**:

   Requires an explicit policy

	Implicit policy $\pi = f(Q_\theta(s,a))$	**Explicit policy** $\pi_\theta(a \vert s)$
Discrete action space	✓	✓
Continuous action space	-	✓

Summary first part

1. **Continuous state space:**

   Small = discretization, large = function approximation

2. **Continuous action space**:

   Requires an explicit policy

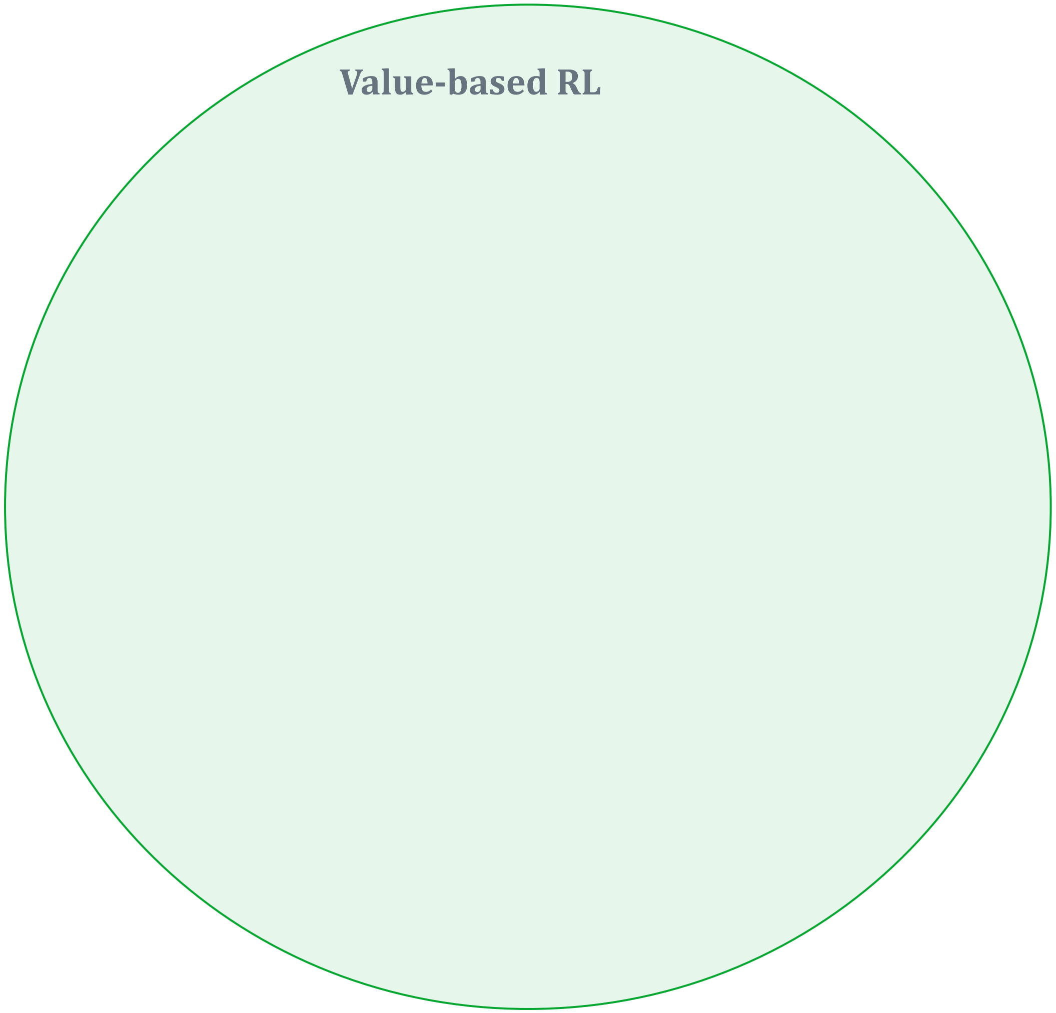
	Implicit policy $\pi = f(Q_\theta(s,a))$	**Explicit policy** $\pi_\theta(a\|s)$
Discrete action space	✓	✓
Continuous action space	-	✓

After break: policy-based RL methods

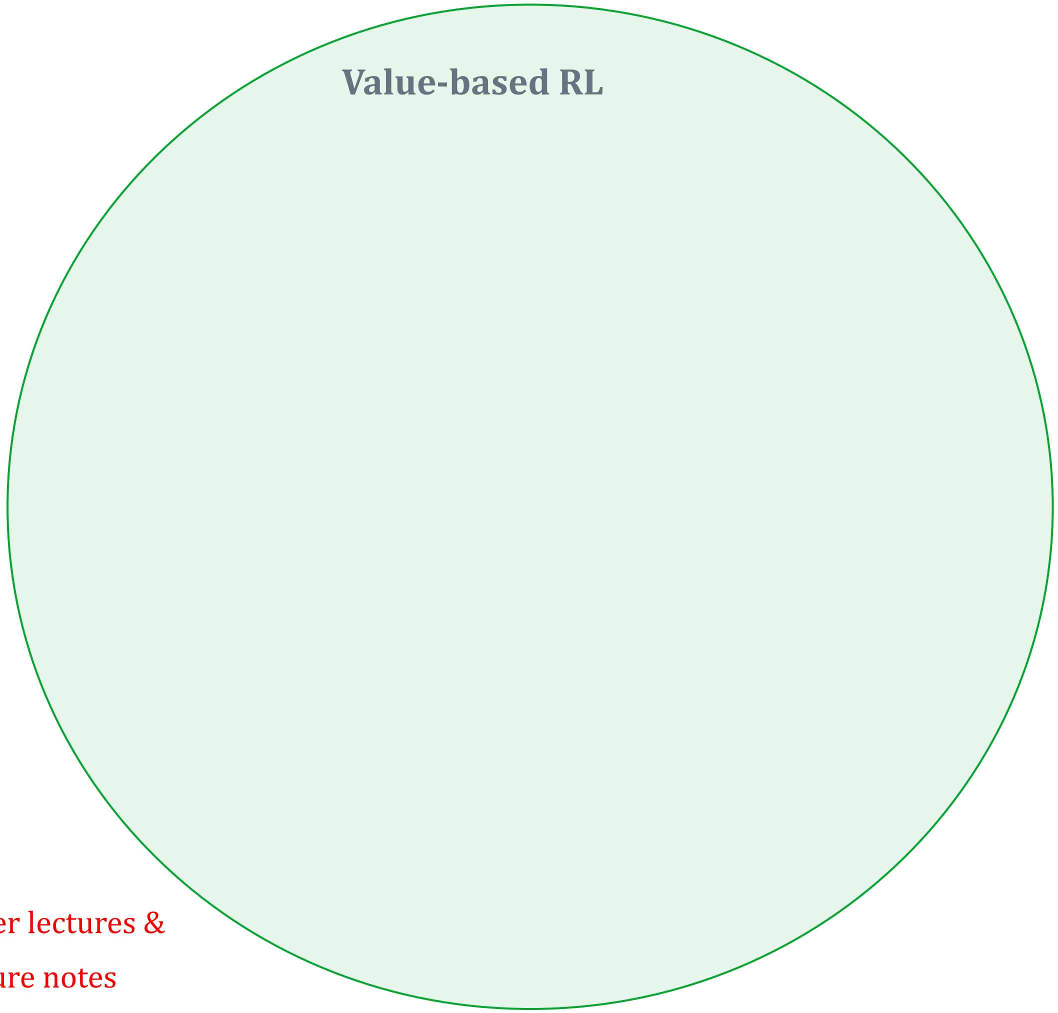
Break

Policy search

Value-based versus policy-based

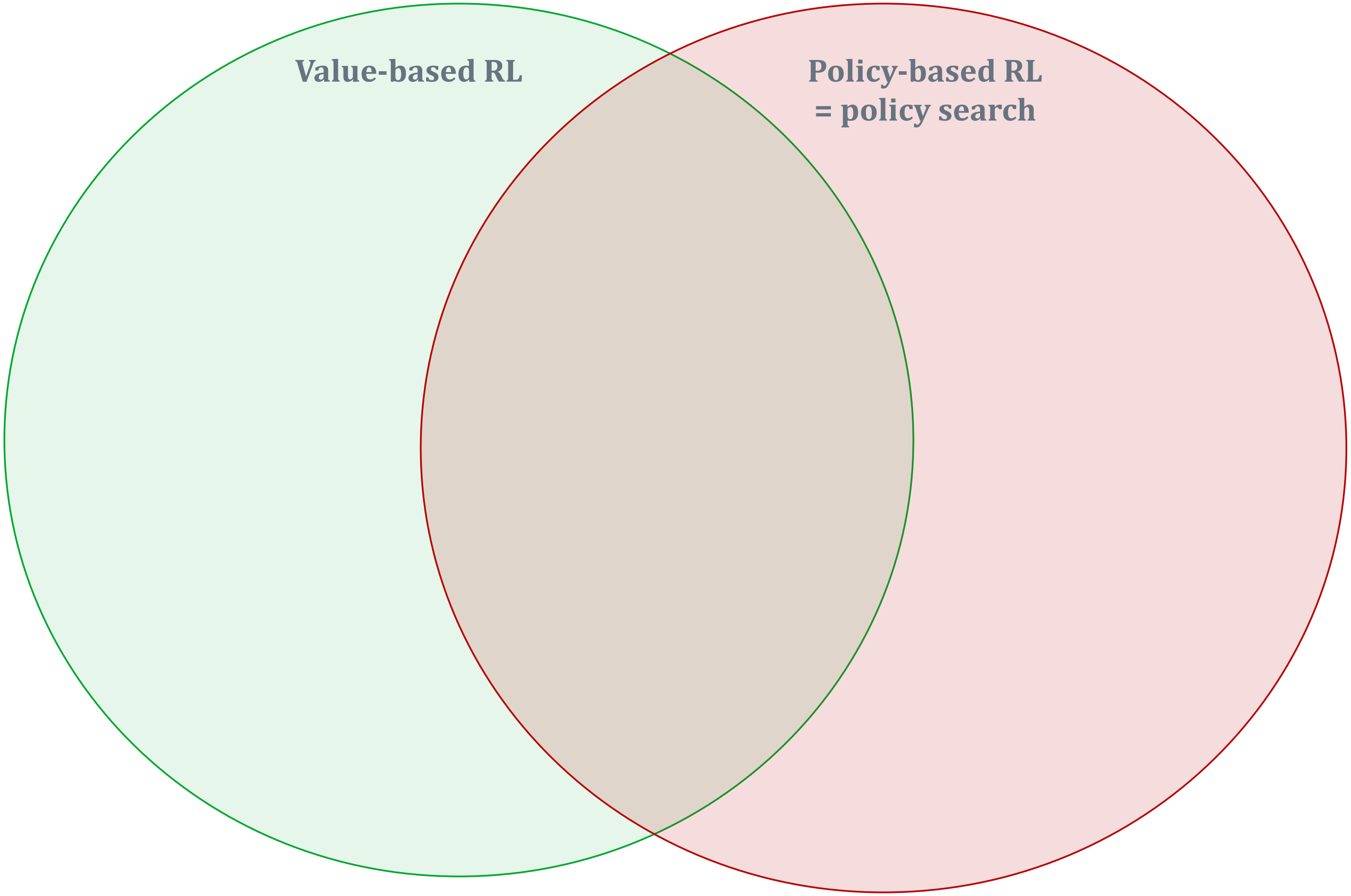


Value-based versus policy-based

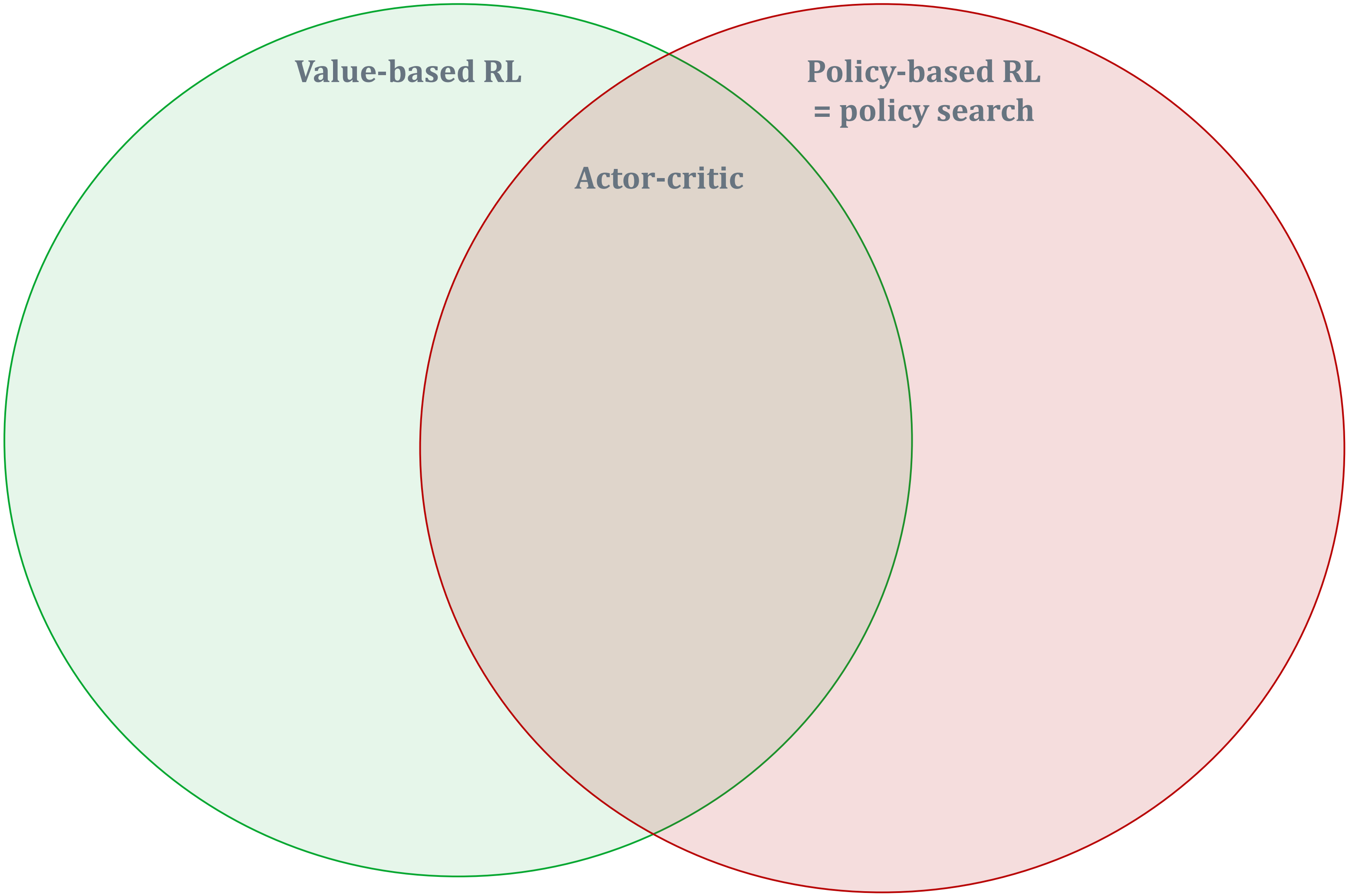


See other lectures &
lecture notes

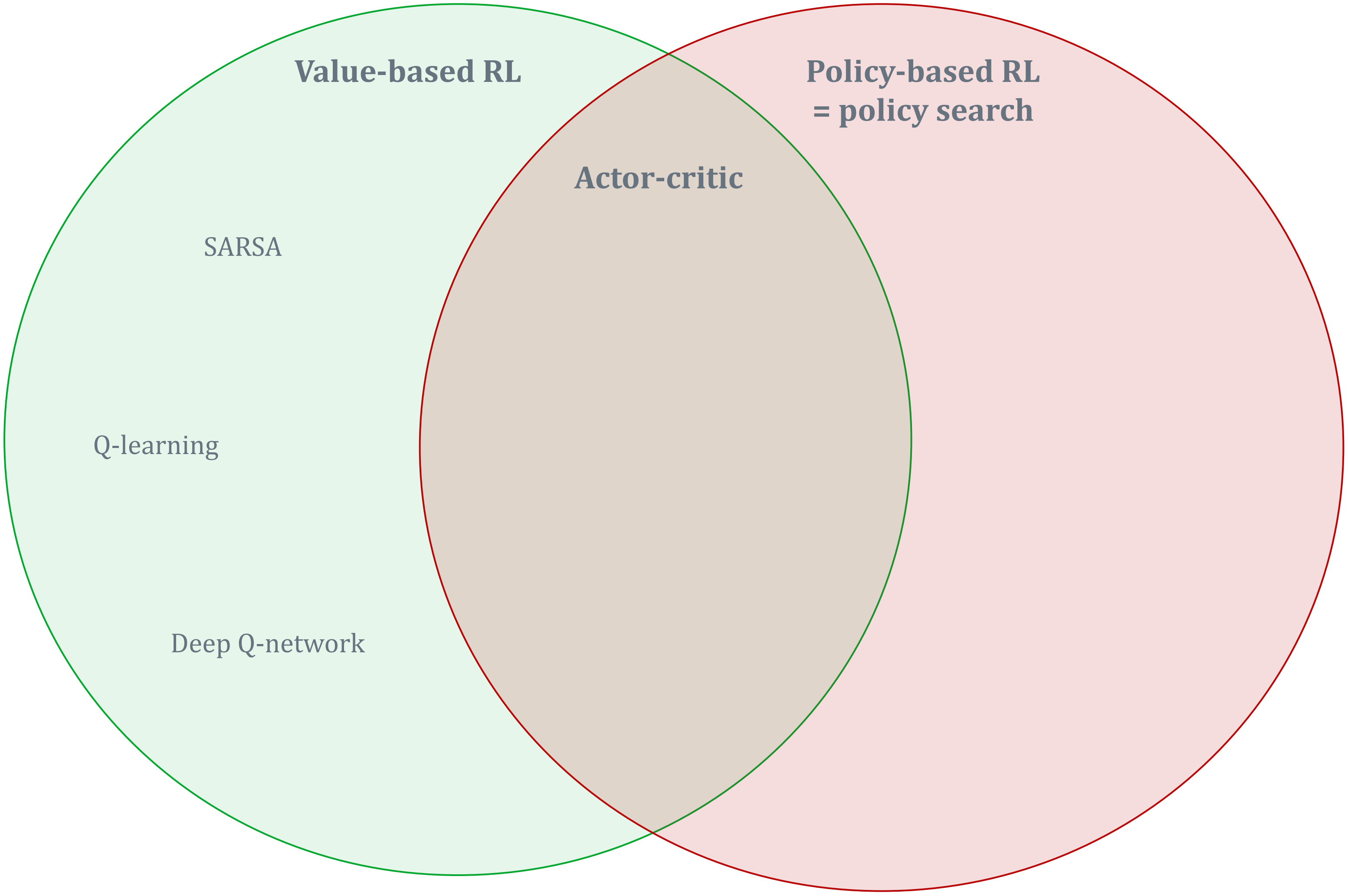
Value-based versus policy-based



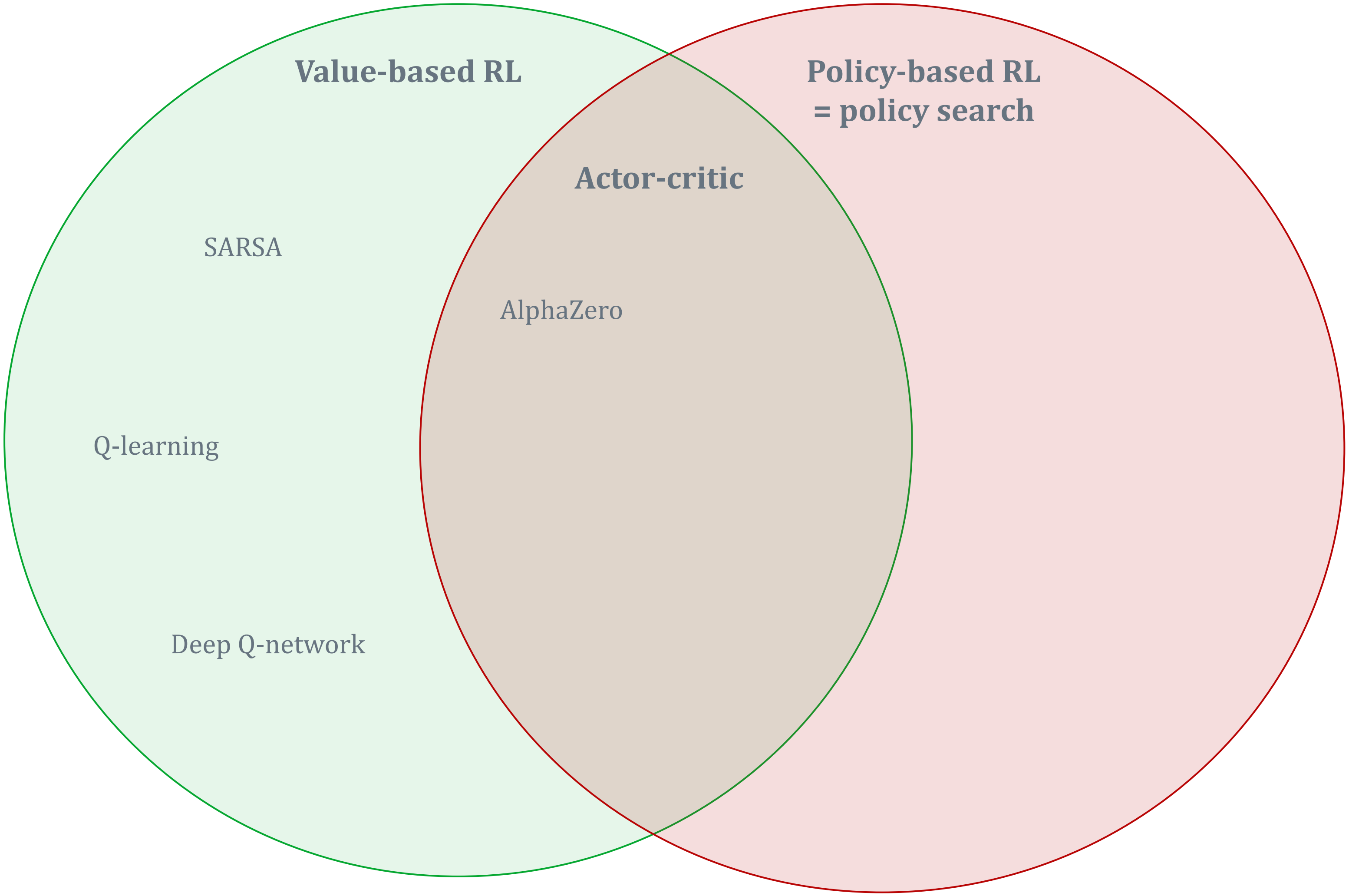
Value-based versus policy-based



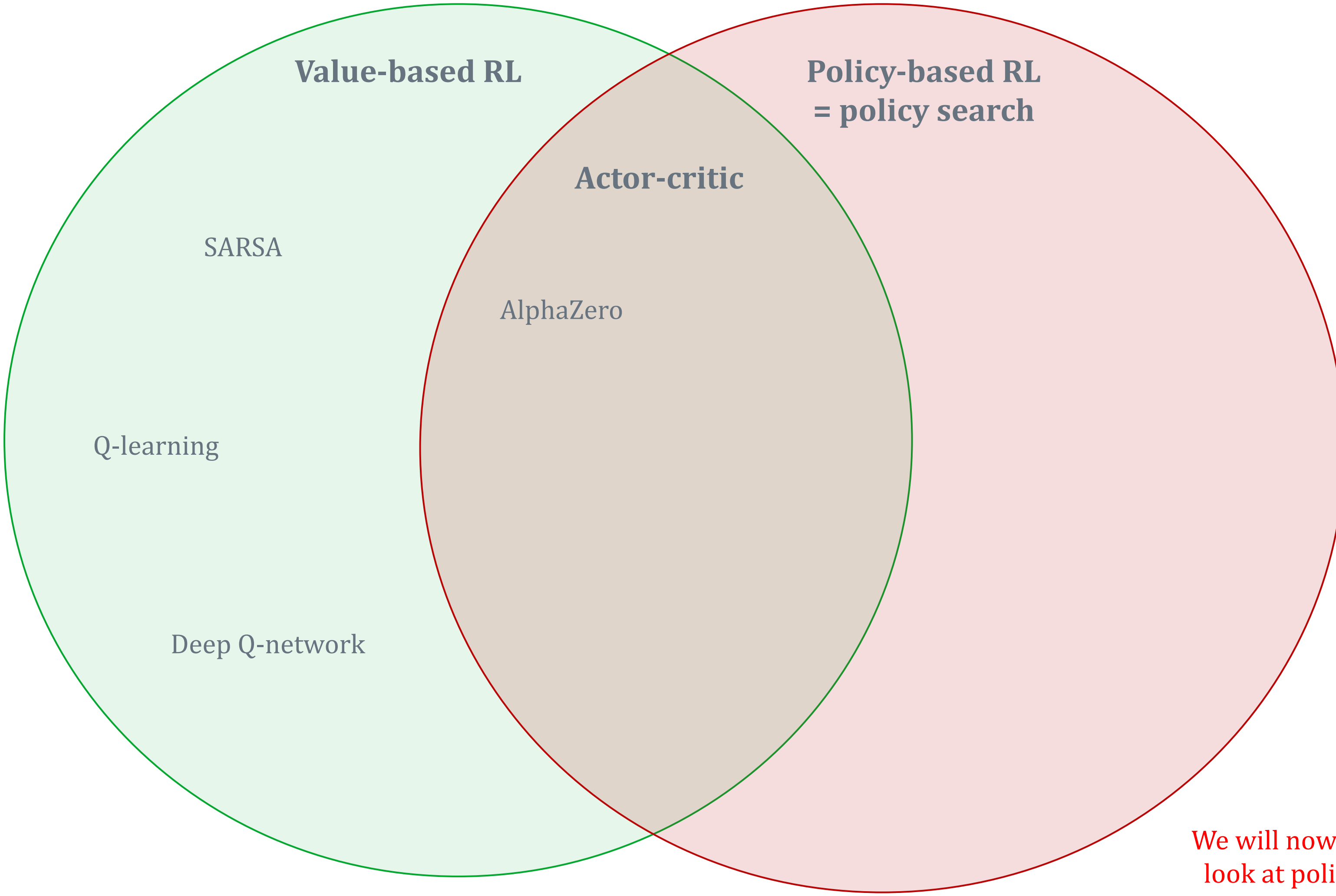
Value-based versus policy-based



Value-based versus policy-based



Value-based versus policy-based



We will now take a closer look at policy-based RL

The reinforcement learning objective

Maximize the expected cumulative return from the start
(w.r.t policy parameters θ)

The reinforcement learning objective

Maximize the expected cumulative return from the start
(w.r.t policy parameters θ)

$$J(\theta) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big]$$

The reinforcement learning objective

Maximize the expected cumulative return from the start
(w.r.t policy parameters θ)

$$J(\theta) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big]$$

Optimization problem
(find optimal policy parameters θ)

The reinforcement learning objective

Maximize the expected cumulative return from the start
(w.r.t policy parameters θ)

$$J(\theta) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big]$$

Optimization problem
(find optimal policy parameters θ)

Gradient-based optimization
(policy gradients)

Gradient-free optimization

The reinforcement learning objective

Maximize the expected cumulative return from the start
(w.r.t policy parameters θ)

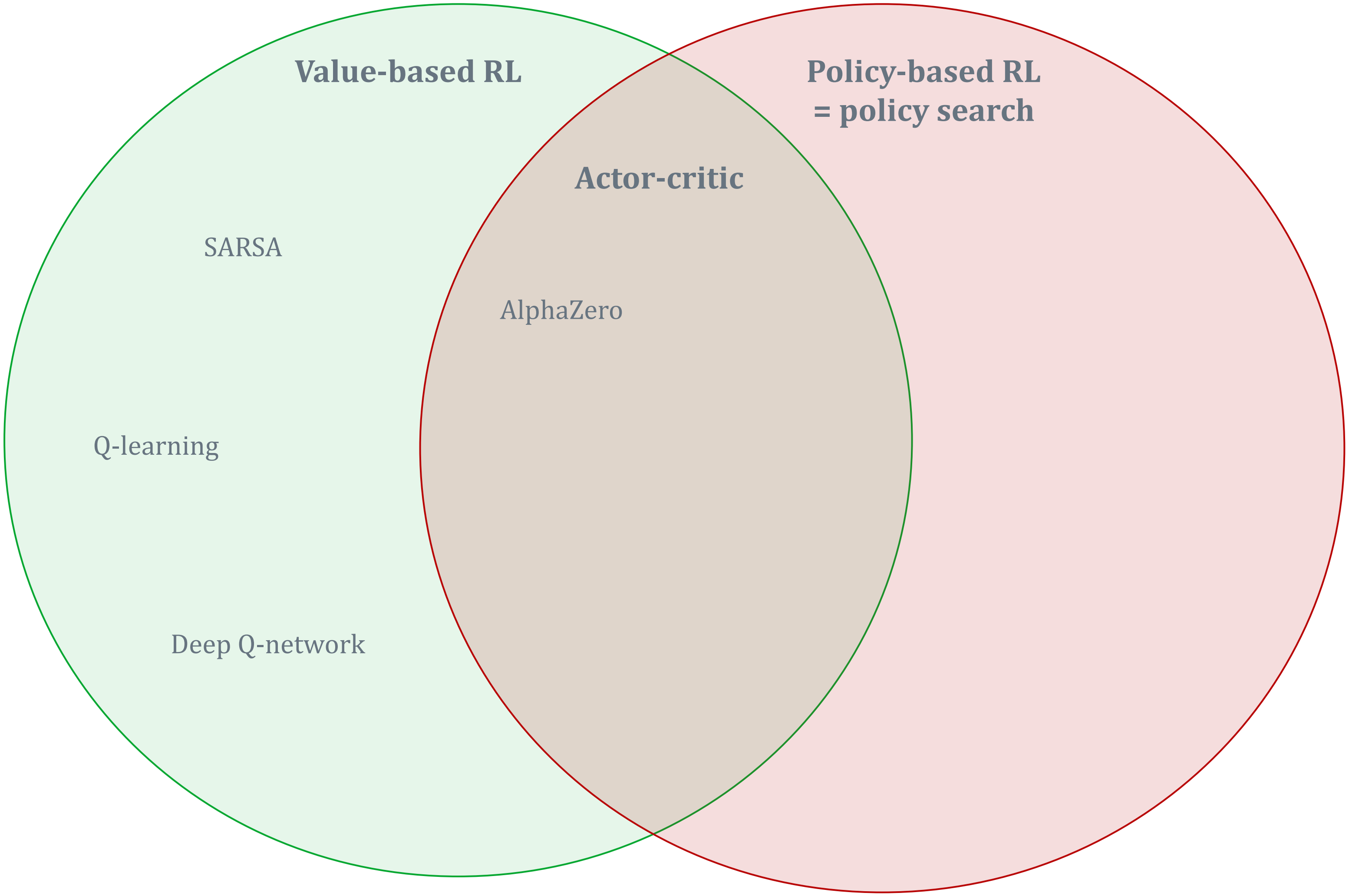
$$J(\theta) = V^{\pi}(s_0) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big]$$

Optimization problem
(find optimal policy parameters θ)

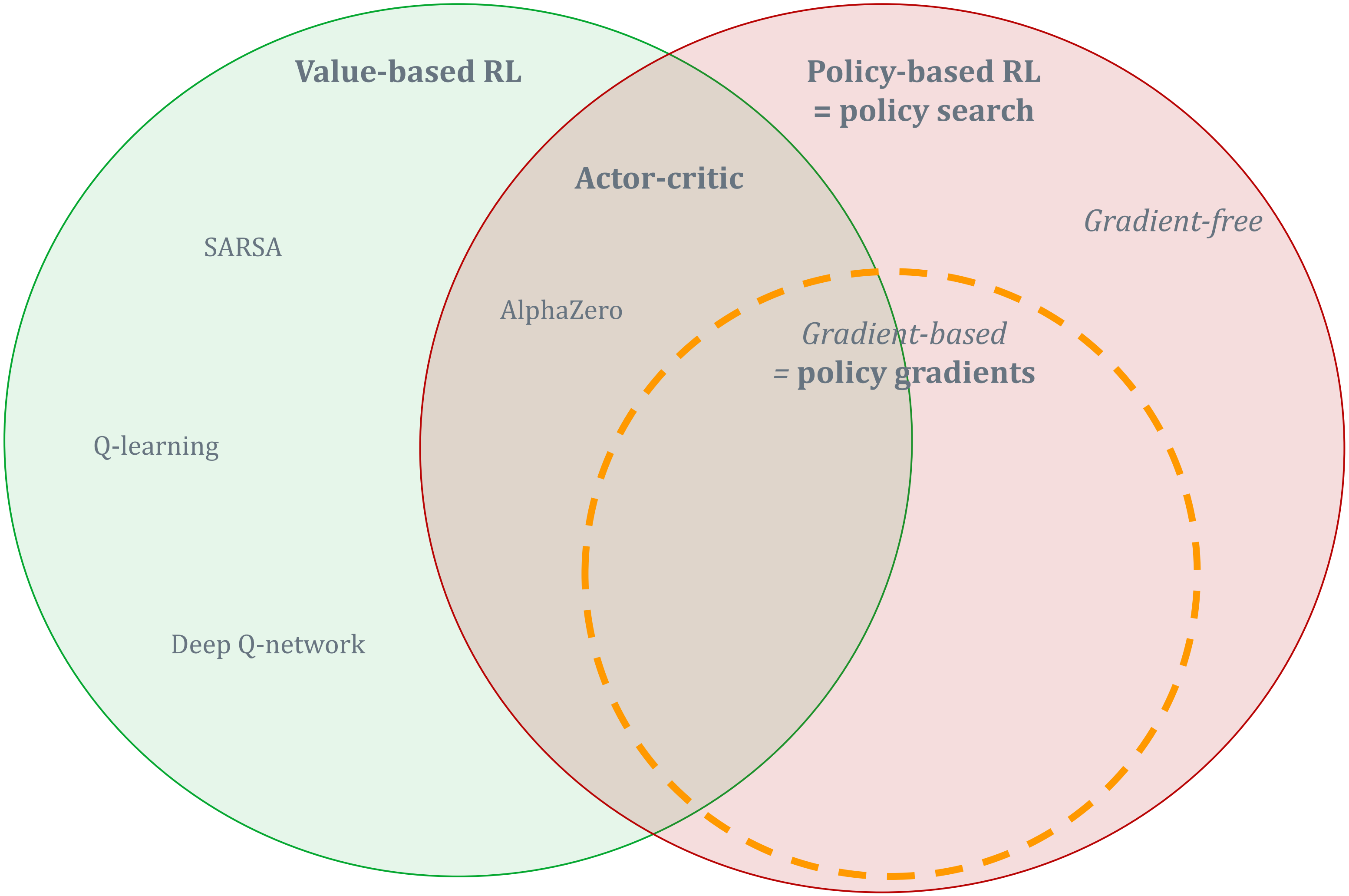
Gradient-based optimization
(policy gradients)

Gradient-free optimization

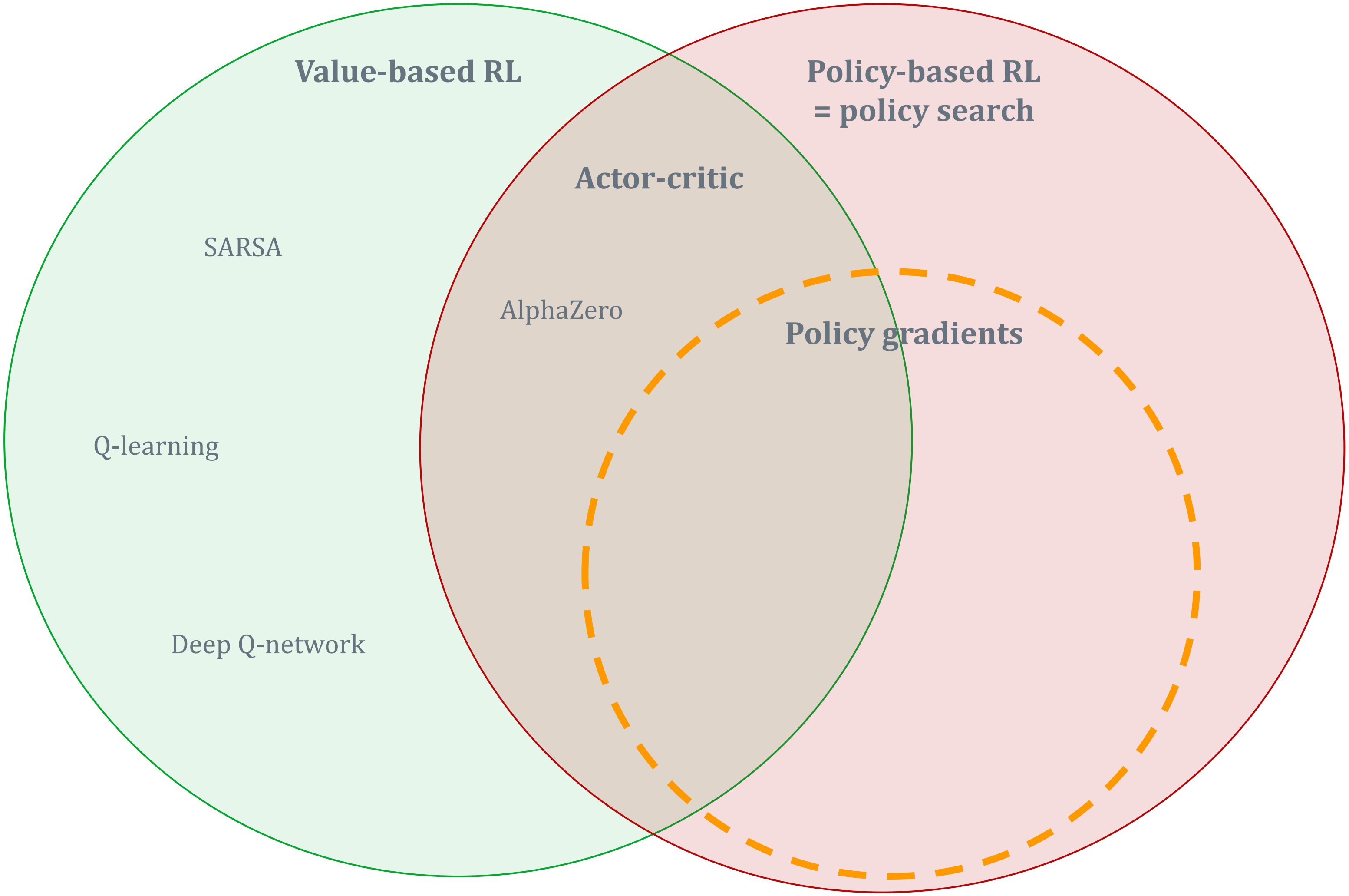
Value-based versus policy-based



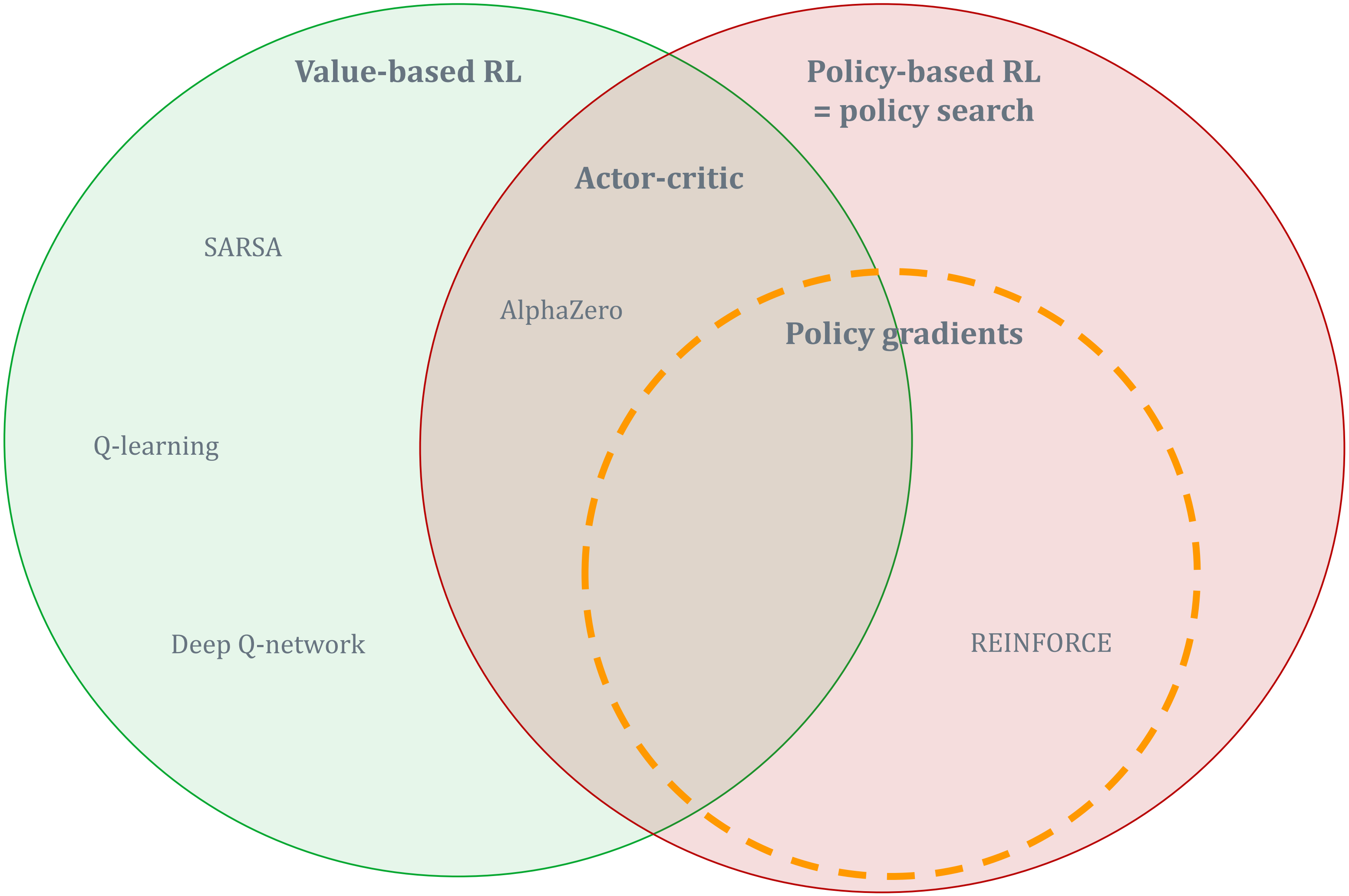
Value-based versus policy-based



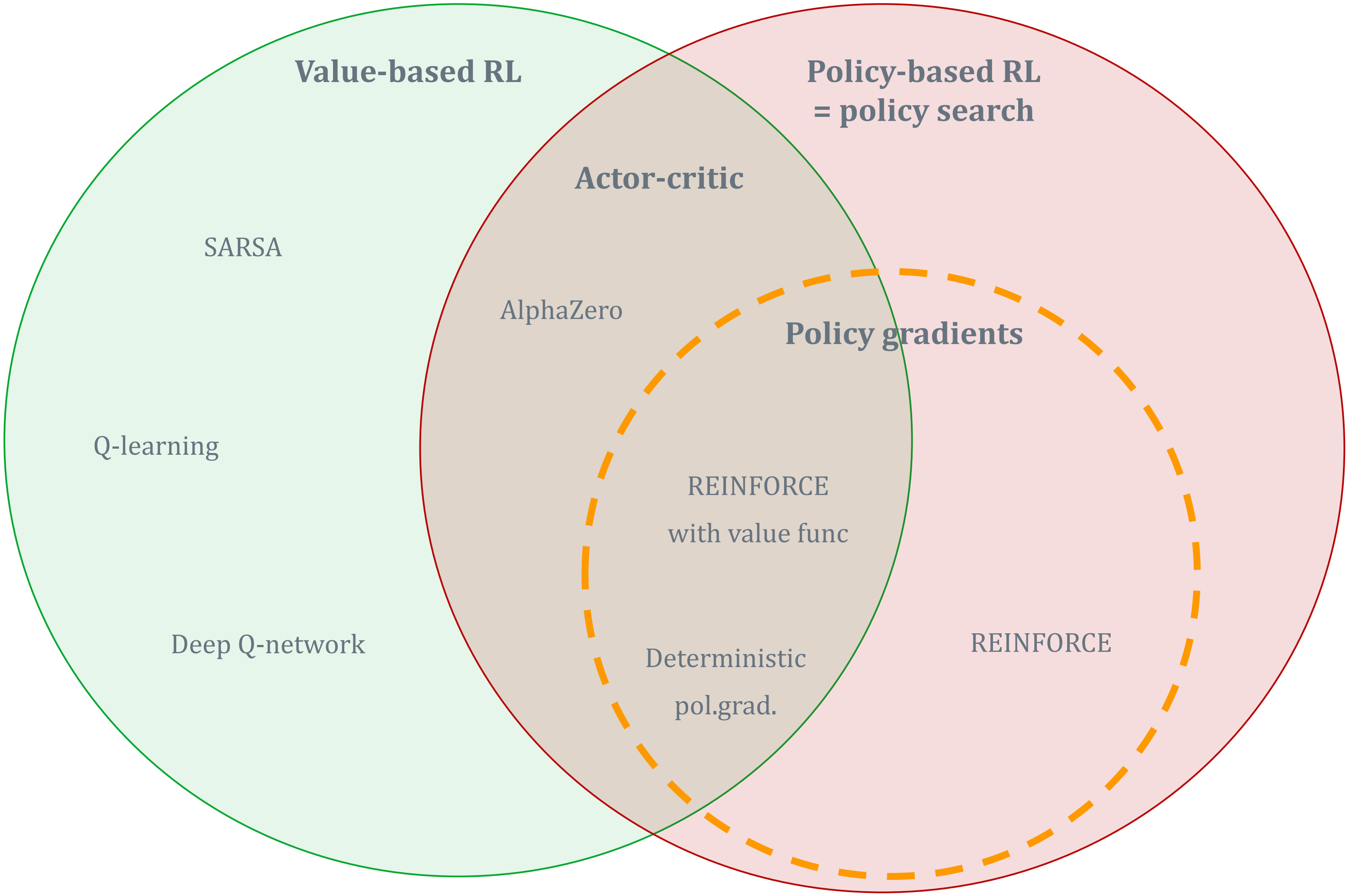
Value-based versus policy-based



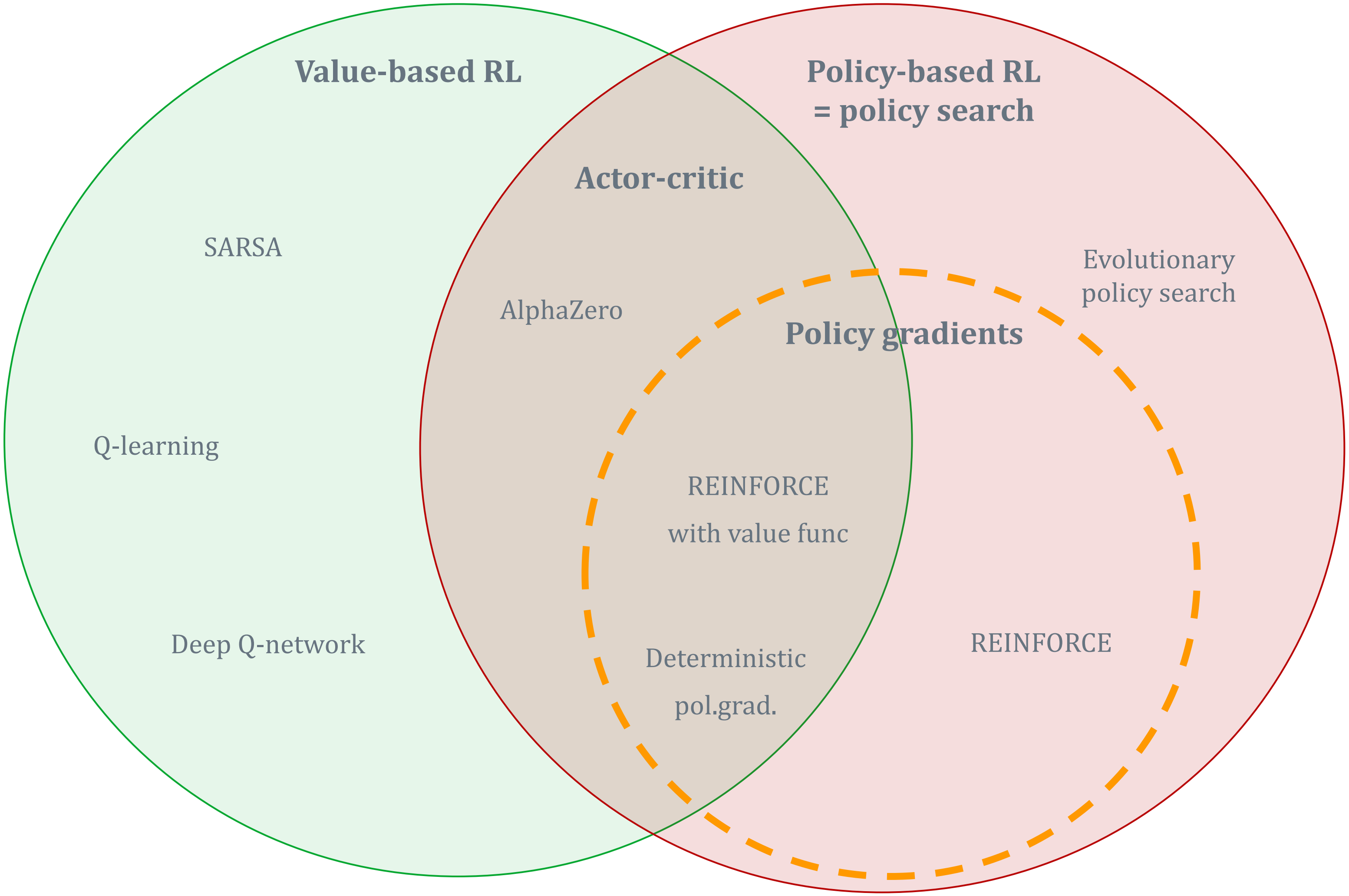
Value-based versus policy-based



Value-based versus policy-based



Value-based versus policy-based



4. Policy gradients

Gradient ascent

Objective

$$\arg\max_{\theta} J(\theta)$$

Gradient ascent
(pseudocode)

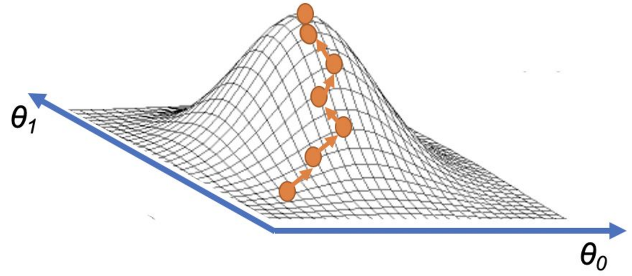
Gradient ascent

Objective

$$\arg\max_{\theta} J(\theta)$$

Gradient ascent
(pseudocode)

repeat
$\quad | \quad \theta \leftarrow \theta + \eta \cdot \nabla_\theta J(\theta)$
until θ *converges*;



Gradient ascent

Objective

$$\arg\max_{\theta} J(\theta)$$

Gradient ascent
(pseudocode)

$$\begin{array}{l}\textbf{repeat}\\ \quad \theta \leftarrow \theta + \eta \cdot \boxed{\nabla_{\theta} J(\theta)}\\ \textbf{until } \theta \ \textit{converges};\end{array}$$

Gradient ascent

Objective

$$\arg\max_{\theta} J(\theta)$$

Gradient ascent
(pseudocode)

repeat
$\quad \theta \leftarrow \theta + \eta \cdot \boxed{\nabla_\theta J(\theta)}$
until θ *converges*;

$$\boxed{\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big]}$$

We will derive the gradient of the expected return w.r.t the policy parameters

Gradient ascent

Objective

$$\arg\max_{\theta} J(\theta)$$

Gradient ascent
(pseudocode)

$$\begin{array}{l}\textbf{repeat}\\ \quad \theta \leftarrow \theta + \eta \cdot \boxed{\nabla_\theta J(\theta)}\\ \textbf{until}\ \theta\ \textit{converges};\end{array}$$

$$\nabla_\theta J(\theta) = \boxed{\nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}}\Big[R(h_0)\Big]$$

We need a derivative of an expectation: appears all over machine learning!

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

We will use the log derivative identity: $\nabla_x \log g(x) = \frac{\nabla_x g(x)}{g(x)}$

(*derivative of log + chain rule*)

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \nabla_\theta \sum_x f(x) \cdot p_\theta(x)$$

definition of expectation

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] &= \nabla_\theta \sum_x f(x) \cdot p_\theta(x) && \text{definition of expectation} \\
&= \sum_x f(x) \cdot \nabla_\theta p_\theta(x) && \text{push gradient through sum}
\end{aligned}$$

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] &= \nabla_\theta \sum_x f(x) \cdot p_\theta(x) && \text{definition of expectation} \\
&= \sum_x f(x) \cdot \nabla_\theta p_\theta(x) && \text{push gradient through sum} \\
&= \sum_x f(x) \cdot p_\theta(x) \cdot \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} && \text{multiply and divide by } p_\theta(x)
\end{aligned}$$

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] &= \nabla_\theta \sum_x f(x) \cdot p_\theta(x) && \text{definition of expectation} \\
&= \sum_x f(x) \cdot \nabla_\theta p_\theta(x) && \text{push gradient through sum} \\
&= \sum_x f(x) \cdot p_\theta(x) \cdot \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} && \text{multiply and divide by } p_\theta(x) \\
&= \sum_x f(x) \cdot p_\theta(x) \cdot \nabla_\theta \log p_\theta(x) && \text{log-derivative rule (Eq. } \boxed{11}\text{)}
\end{aligned}
$$

Differentiate through an expectation

Interested in derivative of expectation: $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)]$

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] &= \nabla_\theta \sum_x f(x) \cdot p_\theta(x) && \text{definition of expectation} \\
&= \sum_x f(x) \cdot \nabla_\theta p_\theta(x) && \text{push gradient through sum} \\
&= \sum_x f(x) \cdot p_\theta(x) \cdot \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} && \text{multiply and divide by } p_\theta(x) \\
&= \sum_x f(x) \cdot p_\theta(x) \cdot \nabla_\theta \log p_\theta(x) && \text{log-derivative rule (Eq. } \boxed{11}\text{)} \\
&= \mathbb{E}_{x \sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)] && \text{rewrite into expectation}
\end{aligned}
$$

Differentiate through an expectation

1. Log derivative trick

$$\boxed{\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] = \mathbb{E}_{x\sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]}$$

Differentiate through an expectation

1. Log derivative trick

$$\boxed{\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \mathbb{E}_{x \sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]}$$

Many other names:

- Reinforcement learning: **REINFORCE**
- Statistics: score function estimator, likelihood ratio method
- Machine learning: black-box variational inference

Differentiate through an expectation

1. Log derivative trick

$$\boxed{\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] = \mathbb{E}_{x\sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]}$$

Many other names:

- Reinforcement learning: **REINFORCE**
- Statistics: score function estimator, likelihood ratio method
- Machine learning: black-box variational inference

2. Reparametrization trick
[not in this course, but e.g. used in variational auto-encoders (deep learning)]

Interpretation of log-derivative trick

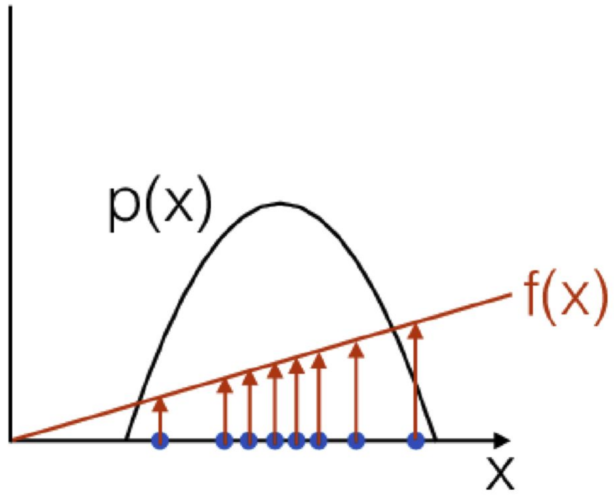
Lot of equations, but what does intuitively happen with this gradient?

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \mathbb{E}_{x \sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]$$

Interpretation of log-derivative trick

Lot of equations, but what does intuitively happen with this gradient?

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \mathbb{E}_{x \sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]$$

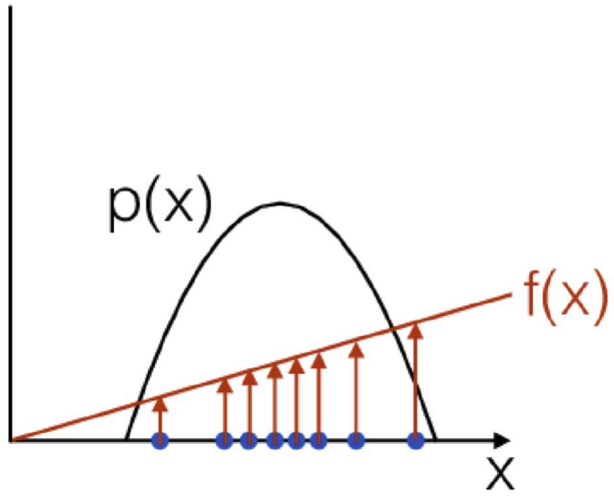


Interpretation of log-derivative trick

Lot of equations, but what does intuitively happen with this gradient?

$$\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] = \mathbb{E}_{x\sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]$$

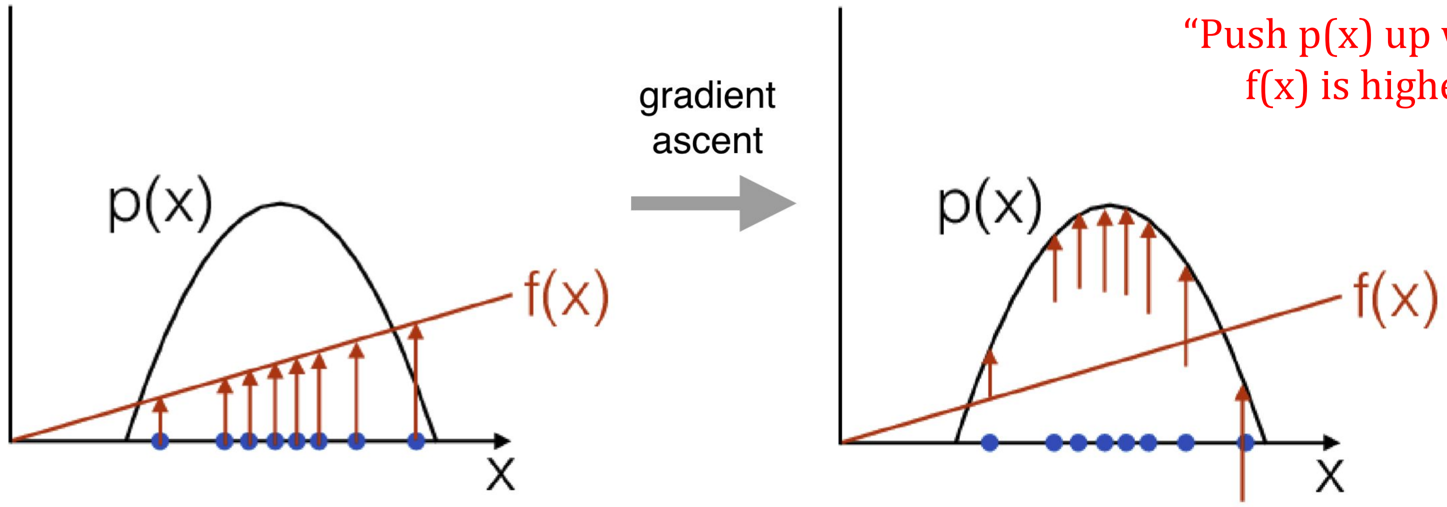
sample x   push up p(x)   with strength f(x)



Interpretation of log-derivative trick

Lot of equations, but what does intuitively happen with this gradient?

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \mathbb{E}_{x \sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]$$

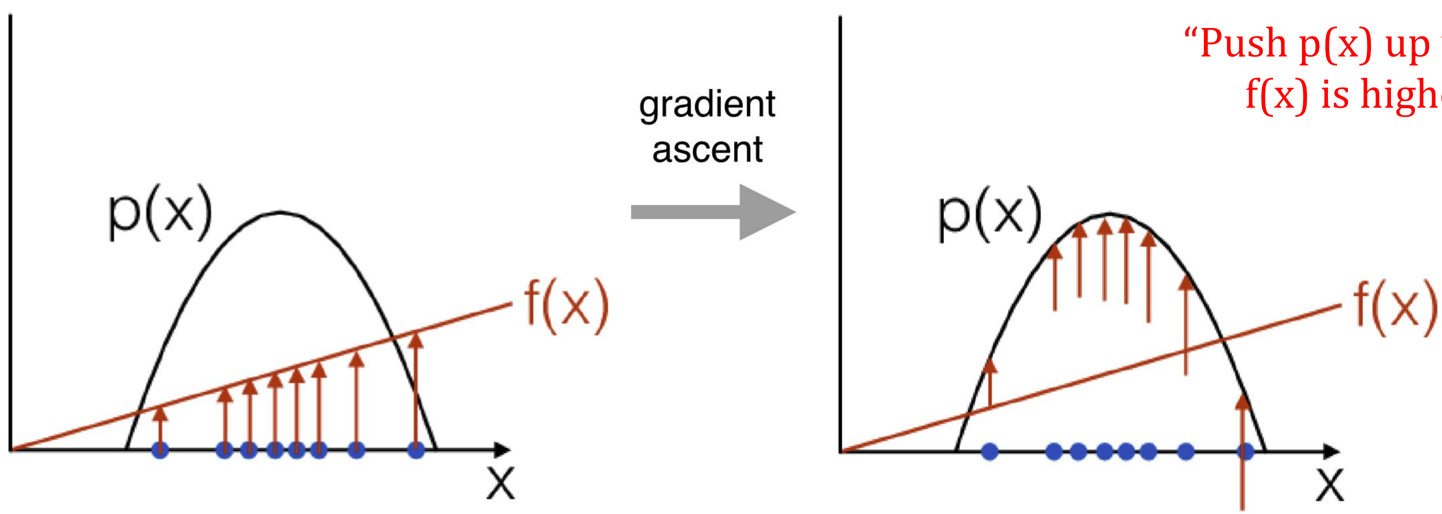


"Push p(x) up where f(x) is higher"

Interpretation of log-derivative trick

Lot of equations, but what does intuitively happen with this gradient?

$$\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] = \mathbb{E}_{x\sim p_\theta(x)}[f(x) \cdot \nabla_\theta \log p_\theta(x)]$$



"Push p(x) up where f(x) is higher"

= "zero-order gradient"

Back to RL objective

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big]$$

Definition of objective

Back to RL objective

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] && \text{Definition of objective} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \nabla_\theta \log p_\theta(h_0)\Big] && \text{Log-derivative trick (Eq. } \boxed{12}\text{)}
\end{aligned}
$$

Simply apply the log-derivative trick!

Back to RL objective

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] \qquad \text{Definition of objective}$$

$$= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \boxed{\nabla_\theta \log p_\theta(h_0)\Big]} \qquad \text{Log-derivative trick (Eq. 12)}$$

Still need to find the derivative of the trace probability

Derivative of trace probability

$$\nabla_\theta \log p_\theta(h_0) = \nabla_\theta \log \left[p_0(s_0) \prod_{t=0}^{n} \pi_\theta(a_t|s_t) \cdot T(s_{t+1}|s_t, a_t) \right] \qquad \text{Definition of } p_\theta(h_0)$$

Derivative of trace probability

Initial state distribution | **Policy probabilities** | **Transition probabilities**

$$\nabla_\theta \log p_\theta(h_0) = \nabla_\theta \log \left[p_0(s_0) \prod_{t=0}^{n} \pi_\theta(a_t|s_t) \cdot T(s_{t+1}|s_t, a_t) \right] \qquad \text{Definition of } p_\theta(h_0)$$

Derivative of trace probability

$$\nabla_\theta \log p_\theta(h_0) = \nabla_\theta \log \left[p_0(s_0) \prod_{t=0}^{n} \pi_\theta(a_t|s_t) \cdot T(s_{t+1}|s_t, a_t) \right] \qquad \text{Definition of } p_\theta(h_0)$$

$$= \nabla_\theta \left[\log p_0(s_0) + \sum_{t=0}^{n} \log \pi_\theta(a_t|s_t) + \sum_{t=0}^{n} \log T(s_{t+1}|s_t, a_t) \right] \qquad \text{Log of product}$$

Derivative of trace probability

$$\nabla_\theta \log p_\theta(h_0) = \nabla_\theta \log \left[p_0(s_0) \prod_{t=0}^{n} \pi_\theta(a_t|s_t) \cdot T(s_{t+1}|s_t, a_t) \right] \qquad \text{Definition of } p_\theta(h_0)$$

$$= \nabla_\theta \left[\cancel{\log p_0(s_0)} + \sum_{t=0}^{n} \log \pi_\theta(a_t|s_t) + \sum_{t=0}^{n} \cancel{\log T(s_{t+1}|s_t, a_t)} \right] \qquad \text{Log of product}$$

Derivative does not depend on (unknown) initial state distribution and transition distribution

Derivative of trace probability

$$\begin{aligned}
\nabla_\theta \log p_\theta(h_0) &= \nabla_\theta \log \left[p_0(s_0) \prod_{t=0}^{n} \pi_\theta(a_t|s_t) \cdot T(s_{t+1}|s_t, a_t) \right] && \text{Definition of } p_\theta(h_0) \\
&= \nabla_\theta \left[\log p_0(s_0) + \sum_{t=0}^{n} \log \pi_\theta(a_t|s_t) + \sum_{t=0}^{n} \log T(s_{t+1}|s_t, a_t) \right] && \text{Log of product} \\
&= \sum_{t=0}^{n} \nabla_\theta \log \pi_\theta(a_t|s_t) && \text{Dependence on } \theta
\end{aligned}$$

Back to RL objective

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] && \text{Definition of objective} \\ &= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\, \boxed{\nabla_\theta \log p_\theta(h_0)}\Big] && \text{Log-derivative trick (Eq. } \boxed{12}\text{)}\end{aligned}$$

Back to RL objective

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] && \text{Definition of objective} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \nabla_\theta \log p_\theta(h_0)\Big] && \text{Log-derivative trick (Eq. } \boxed{12}\text{)} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \sum_{t=0}^{n} \nabla_\theta \log \pi_\theta(a_t|s_t)\Big] && \text{Gradient of trace (Eq. } \boxed{13}\text{)}
\end{aligned}$$

Back to RL objective

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] && \text{Definition of objective} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \nabla_\theta \log p_\theta(h_0)\Big] && \text{Log-derivative trick (Eq. } \boxed{12}\text{)} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \sum_{t=0}^{n} \nabla_\theta \log \pi_\theta(a_t|s_t)\Big] && \text{Gradient of trace (Eq. } \boxed{13}\text{)}
\end{aligned}
$$

Return of the entire trace

Sum over each timestep in the trace

Derivative of log policy

Back to RL objective

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] && \text{Definition of objective} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \nabla_\theta \log p_\theta(h_0)\Big] && \text{Log-derivative trick (Eq. } \boxed{12}\text{)} \\
&= \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0) \cdot \sum_{t=0}^{n} \nabla_\theta \log \pi_\theta(a_t|s_t)\Big] && \text{Gradient of trace (Eq. } \boxed{13}\text{)}
\end{aligned}
$$

Makes sense to move $R(h_0)$ inside the sum,
since return of action only depends on future rewards

Policy gradient theorem

a.k.a. **REINFORCE**

Policy gradient theorem

a.k.a. **REINFORCE**

$$\boxed{\nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[\sum_{t=0}^{n} R(h_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]}$$

Policy gradient theorem

a.k.a. **REINFORCE**

$$\nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[\sum_{t=0}^{n} R(h_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]$$

When you use automatic differentiation software, like Tensorflow or Pytorch, you would implement the following loss:

$$L(\theta) = -\frac{1}{M}\sum_{i=1}^{M}\Big[\sum_{t=0}^{n} R(h_t^i) \log \pi_\theta(a_t|s_t)\Big]$$

Policy gradient theorem

a.k.a. **REINFORCE**

$$\nabla_\theta \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[R(h_0)\Big] = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\Big[\sum_{t=0}^{n} R(h_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]$$

When you use automatic differentiation software, like Tensorflow or Pytorch, you would implement the following loss:

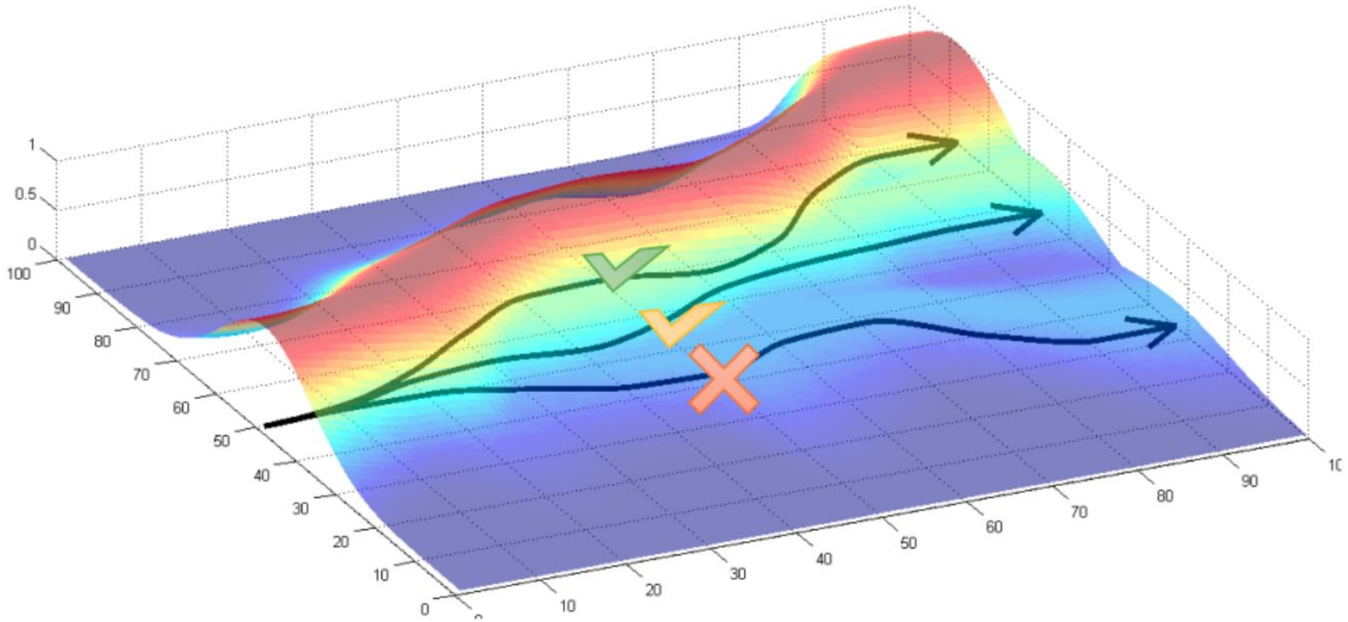
$$L(\theta) = -\frac{1}{M} \sum_{i=1}^{M} \Big[\sum_{t=0}^{n} R(h_t^i) \log \pi_\theta(a_t|s_t)\Big]$$

Minus since loss gets minimized

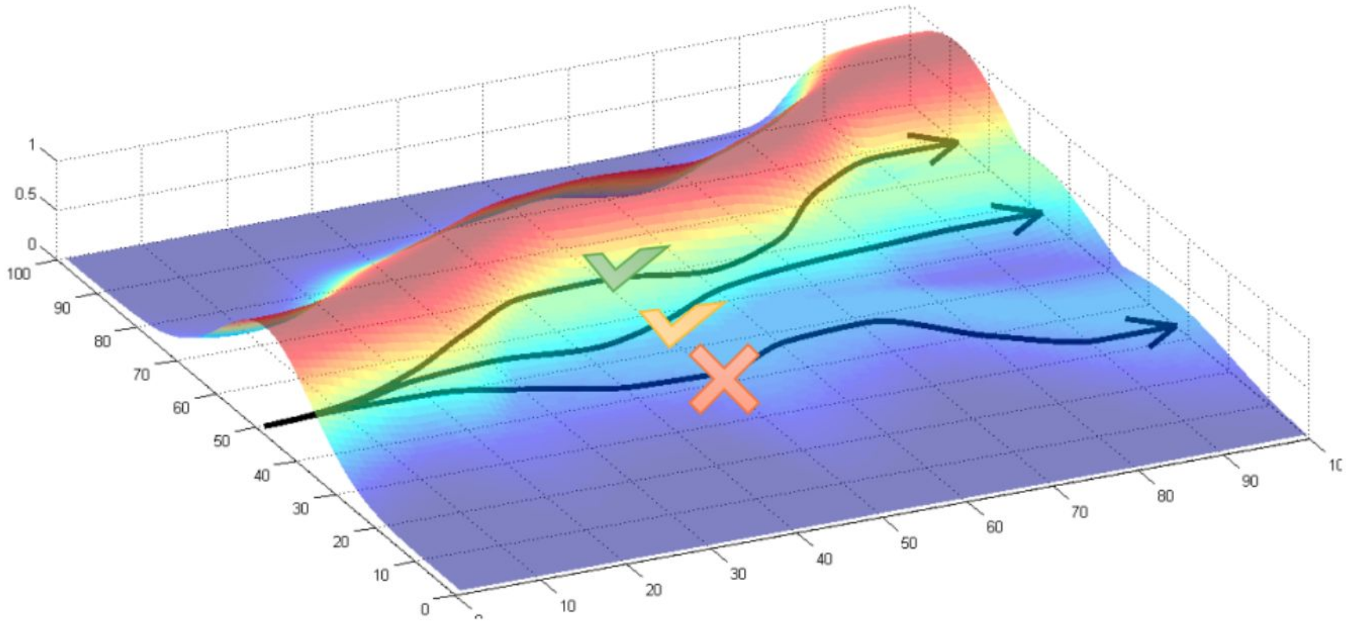
Sample M traces

Derivative of this expression is the policy gradient

Interpretation of policy gradient

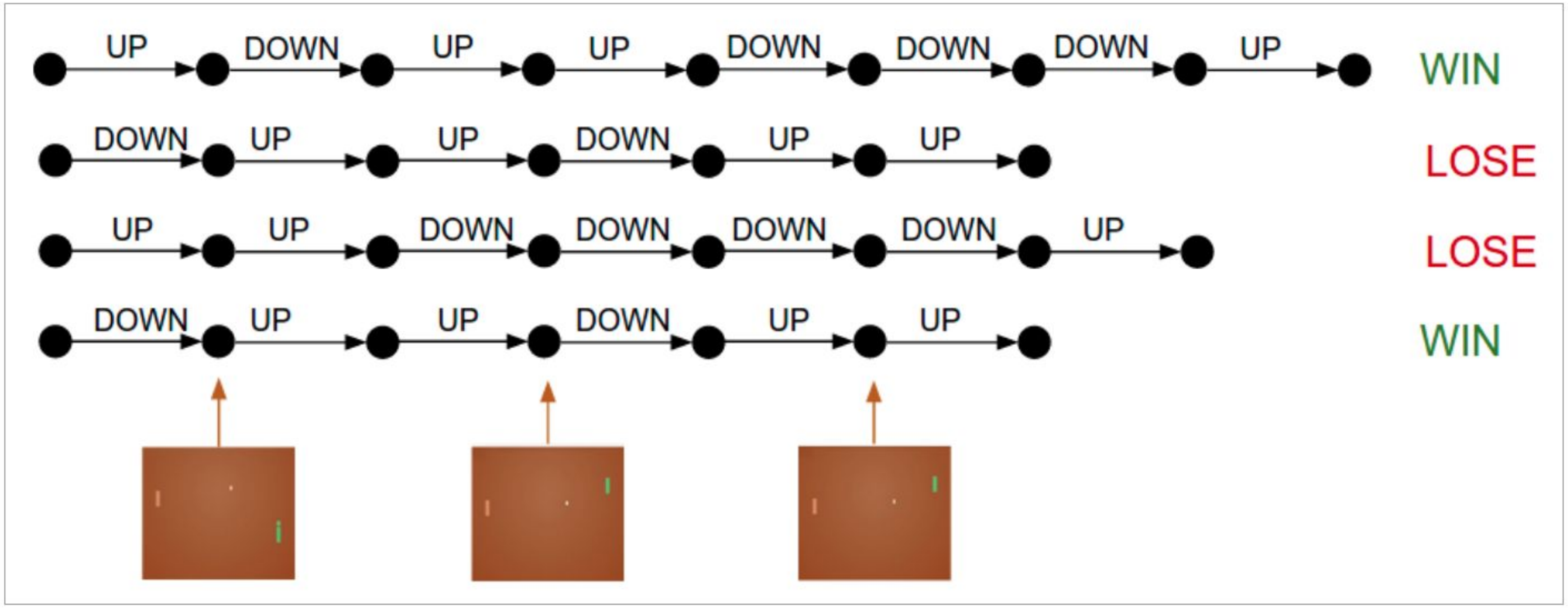


Interpretation of policy gradient

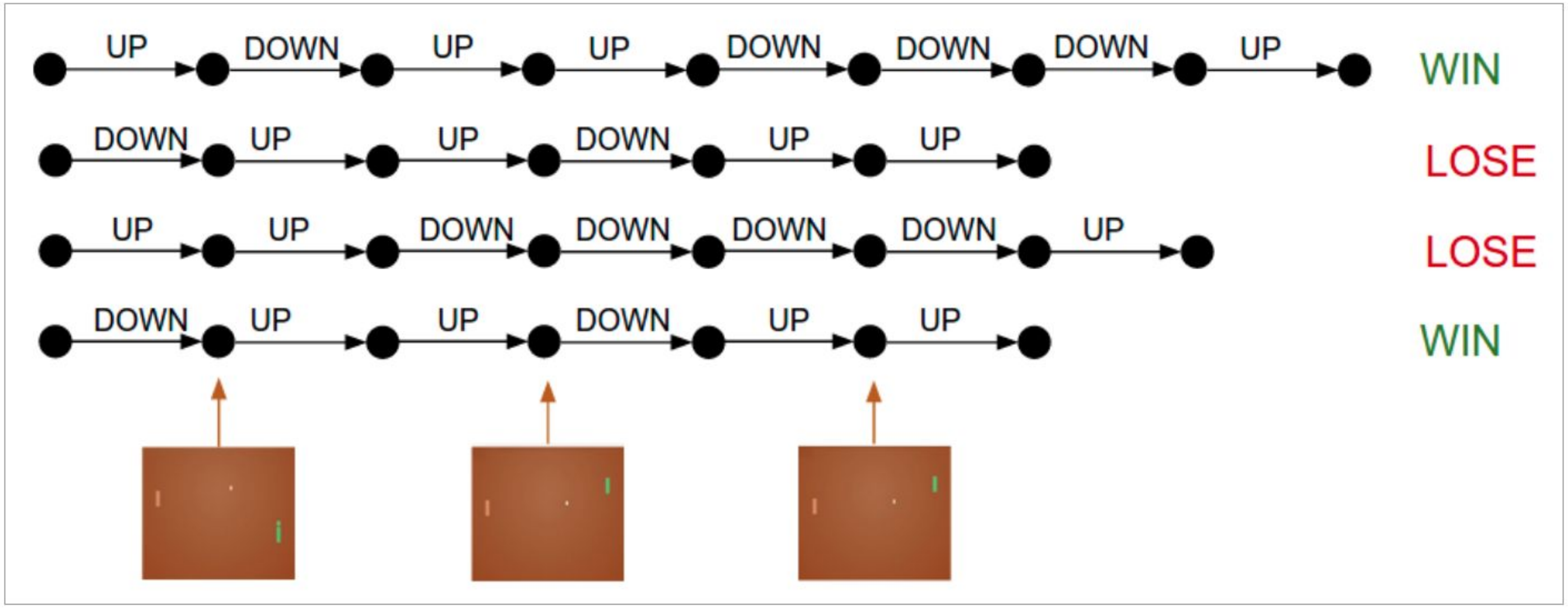


You “reinforce” the actions which give good returns

Interpretation of policy gradient



Interpretation of policy gradient



"Encourage each state-action pair in the successful traces"

"Discourage each state-action pair in the unsuccessful traces"

Exploration in policy gradient

Deterministic policy:

Stochastic policy:

Exploration in policy gradient

Deterministic policy:

- Add noise, e.g., Gaussian: $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(s) + \mathcal{N}(0, \sigma)$$

Stochastic policy:

Exploration in policy gradient

Deterministic policy:

- Add noise, e.g., Gaussian:

$$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(s) + \mathcal{N}(0, \sigma)$$

Stochastic policy:

- Simply sample from policy

$$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(a|s)$$

Exploration in policy gradient

Deterministic policy:

- Add noise, e.g., Gaussian: $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(s) + \mathcal{N}(0, \sigma)$$

Stochastic policy:

- Simply sample from policy $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(a|s)$$ **Risk of collapse!**

Exploration in policy gradient

Deterministic policy:

- Add noise, e.g., Gaussian: $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(s) + \mathcal{N}(0,\sigma)$$

Stochastic policy:

- Simply sample from policy $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(a|s)$$ **Risk of collapse!**

- Add entropy regularization $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(a|s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\left[\sum_{t=0}^{n} R_t \nabla_\theta \log \pi_\theta(a_t|s_t) + \eta \nabla_\theta H[\pi_\theta(a|s)]\right]$$

Exploration in policy gradient

Deterministic policy:

- Add noise, e.g., Gaussian: $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(s) + \mathcal{N}(0,\sigma)$$

Stochastic policy:

- Simply sample from policy $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(a|s)$$ **Risk of collapse!**

- Add entropy regularization $$\pi_{\theta,\text{behaviour}}(a|s) = \pi_\theta(a|s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\left[\sum_{t=0}^{n} R_t \nabla_\theta \log \pi_\theta(a_t|s_t) + \eta \nabla_\theta H[\pi_\theta(a|s)]\right]$$

Optimize return but trade-off against high-entropy (broad) policy

Monte Carlo Policy Gradient

Algorithm 3: Monte Carlo policy gradient (REINFORCE)

Input: A differentiable policy $\pi_\theta(a|s)$, parametrized by $\theta \in \mathbb{R}^d$.
A learning rate η.
Initialization: Randomly initialize θ in $\mathbb{R}^d$.
while *not converged* **do**
  grad $\leftarrow 0$
  for $m \in 1, .., M$ **do**
    Sample trace $h_0 = \{s_0, a_0, r_0, s_1, .., s_{n+1}\}$ following $\pi_\theta(a|s)$
    $R \leftarrow 0$
    for $t \in n, .., 1, 0$ **do**
      $R \leftarrow r_t + \gamma \cdot R$ `/* Backwards through trace */`
      grad $+= R \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)$ `/* Add to total gradient */`
    end
  end
  $\theta \leftarrow \theta + \eta \cdot$ **grad**
end
Return $\pi_\theta(a|s)$

Algorithm in lecture notes!

5. Actor-critic

Actor-critic

Use a value function to potentially get a better policy gradient update

Actor-critic

Use a value function to potentially get a better policy gradient update

1. Within policy gradient theorem:

Actor-critic

Use a value function to potentially get a better policy gradient update

1. <u>Within policy gradient theorem</u>:

    a. Bootstrapping   lower variance in cumulative reward estimate

Actor-critic

Use a value function to potentially get a better policy gradient update

1. Within policy gradient theorem:

    a. Bootstrapping — lower variance in cumulative reward estimate

    b. Baseline subtraction — lower variance in gradient estimate

Actor-critic

Use a value function to potentially get a better policy gradient update

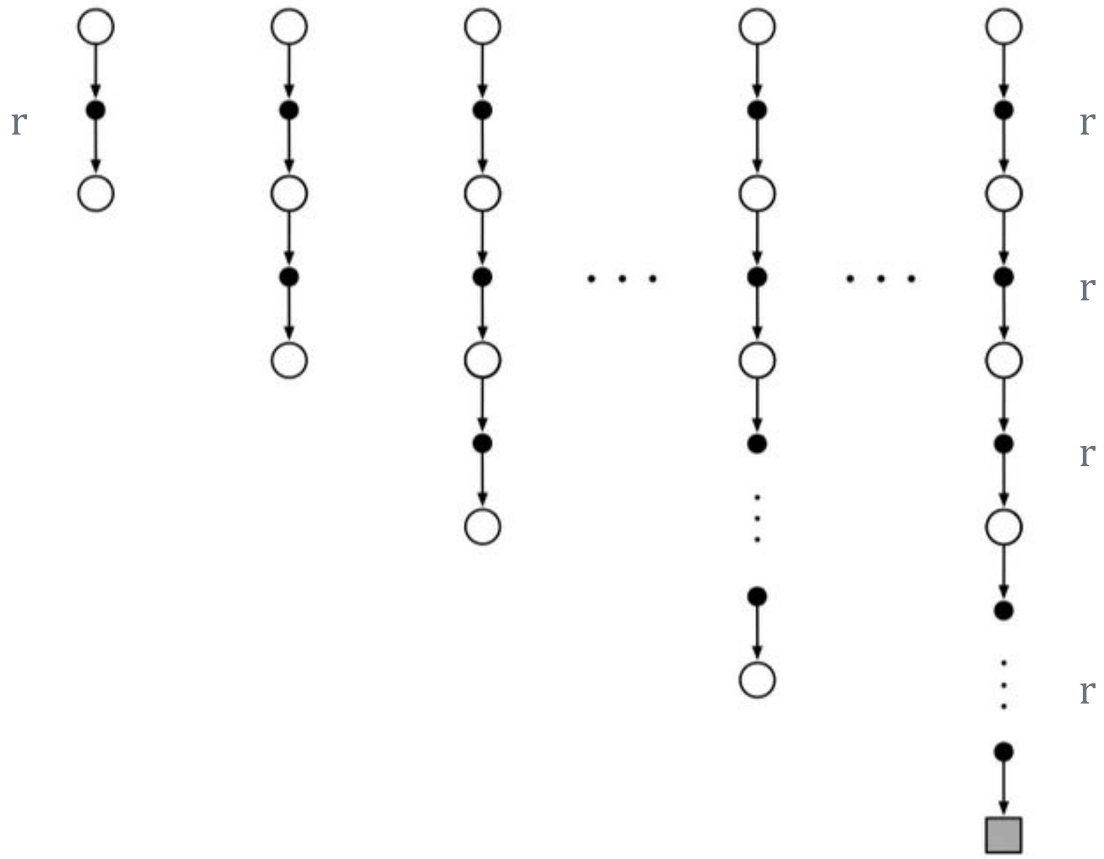
1. Within policy gradient theorem:
   a. Bootstrapping — lower variance in cumulative reward estimate
   b. Baseline subtraction — lower variance in gradient estimate
2. Other type of update:
   a. Deterministic policy gradient

Actor-critic

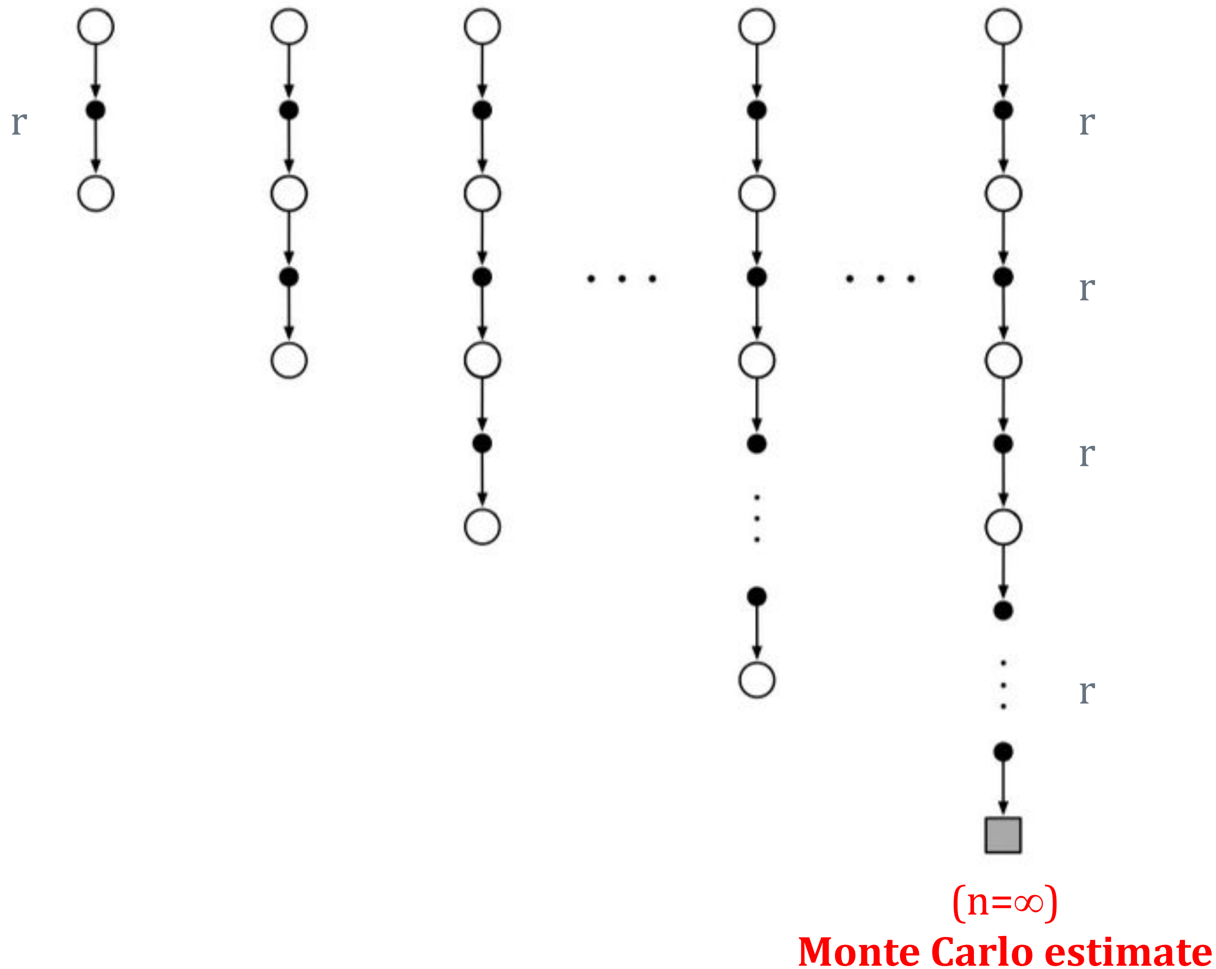
Use a value function to potentially get a better policy gradient update

1. Within policy gradient theorem:
   a. Bootstrapping — lower variance in cumulative reward estimate
   b. Baseline subtraction — lower variance in gradient estimate
2. Other type of update:
   a. Deterministic policy gradient

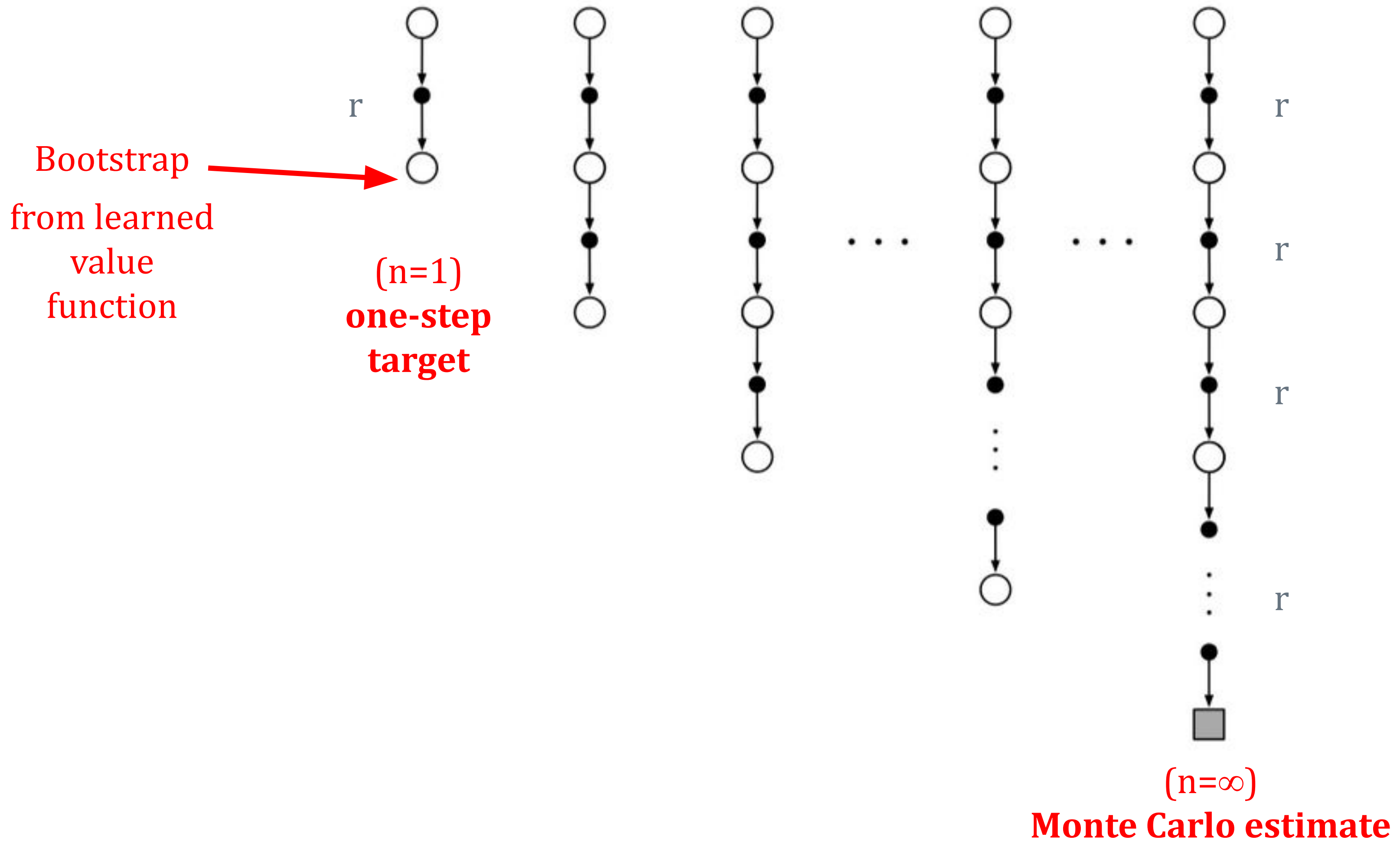
Estimating cumulative reward



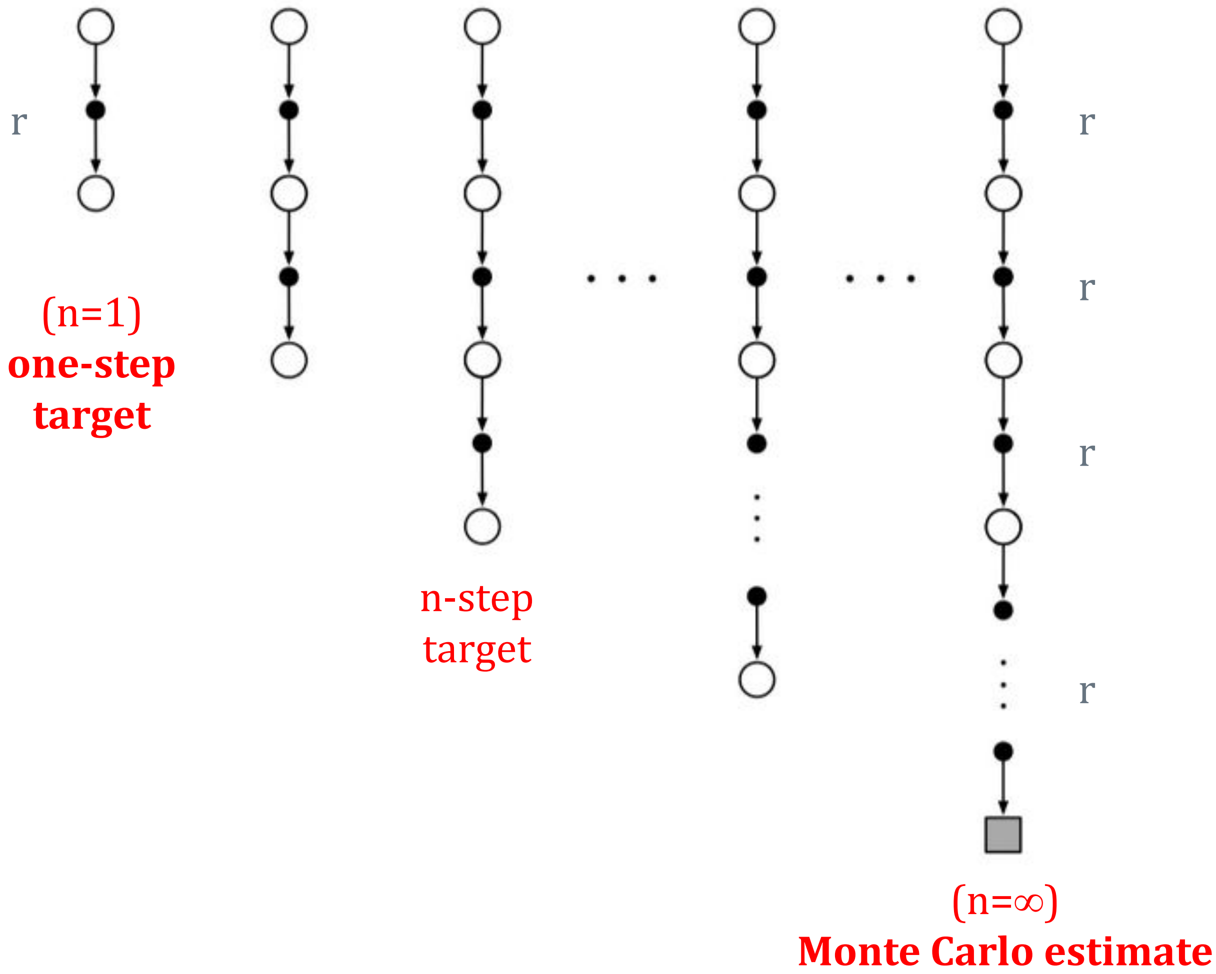
Estimating cumulative reward



Estimating cumulative reward



Estimating cumulative reward



Estimating cumulative reward

General formulation of cumulative reward estimation

$$\hat{Q}_{\text{n-step}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\theta(s_{t+n})$$

Estimating cumulative reward

General formulation of cumulative reward estimation

$$\hat{Q}_{\text{n-step}}(s_t, a_t) = \sum_{k=0}^{\boxed{n}-1} r_{t+k} + V_\theta(s_{t+n})$$

Estimating cumulative reward

General formulation of cumulative reward estimation

$$\hat{Q}_{\text{n-step}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + \boxed{V_\theta(s_{t+n})}$$

bootstrap

(unless n=∞)

Estimating cumulative reward

General formulation of cumulative reward estimation

$$\hat{Q}_{\text{n-step}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + \boxed{V_\theta(s_{t+n})}$$

Off-policy

On-policy

Estimating cumulative reward

General formulation of cumulative reward estimation

$$\hat{Q}_{\text{n-step}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + \boxed{V_\theta(s_{t+n})}$$

<u>Off-policy</u>

<u>On-policy</u>

$$V_\theta(s) = \max_a Q_\theta(s, a)$$

and

$$n = 1$$

Estimating cumulative reward

General formulation of cumulative reward estimation

$$\hat{Q}_{\text{n-step}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + \boxed{V_\theta(s_{t+n})}$$

Off-policy

$$V_\theta(s) = \max_a Q_\theta(s, a)$$

and

$$n = 1$$

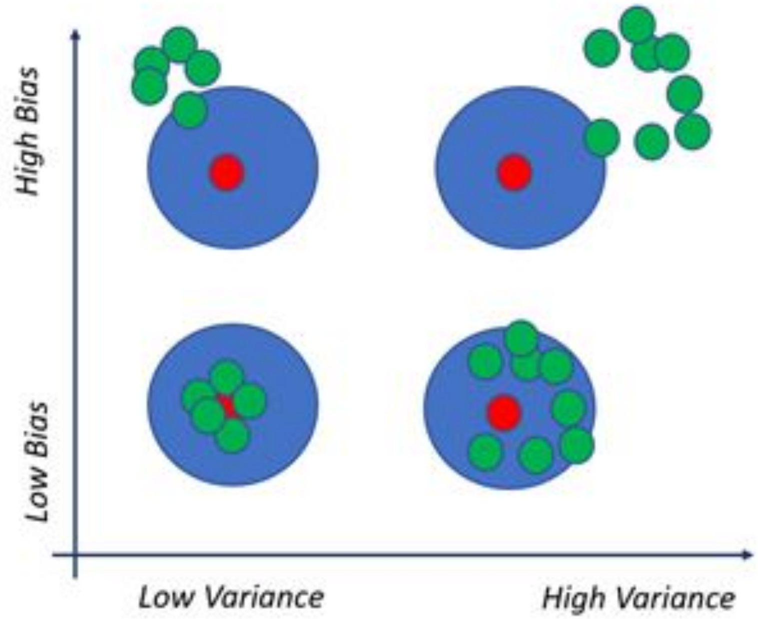
On-policy

$$V_\theta(s) = Q_\theta(s, a) \text{ for } a \sim \pi_\theta(a|s)$$

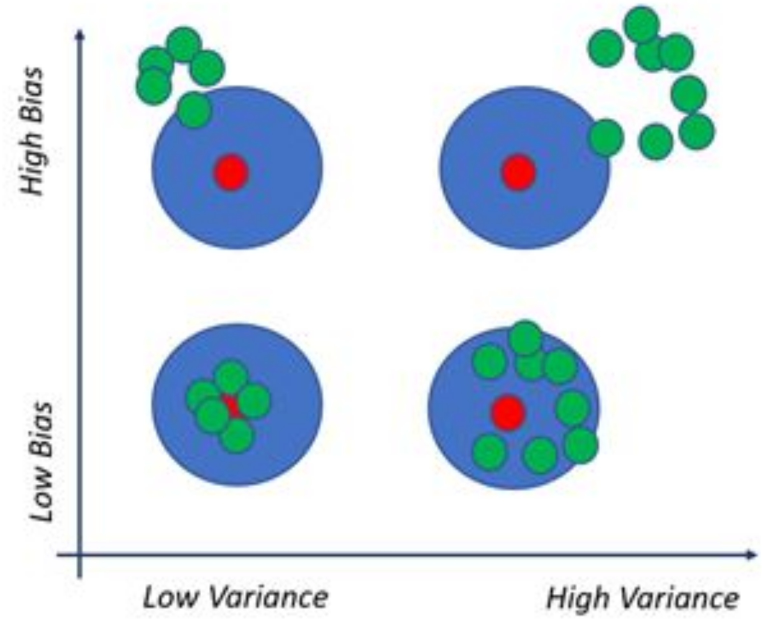
and/or

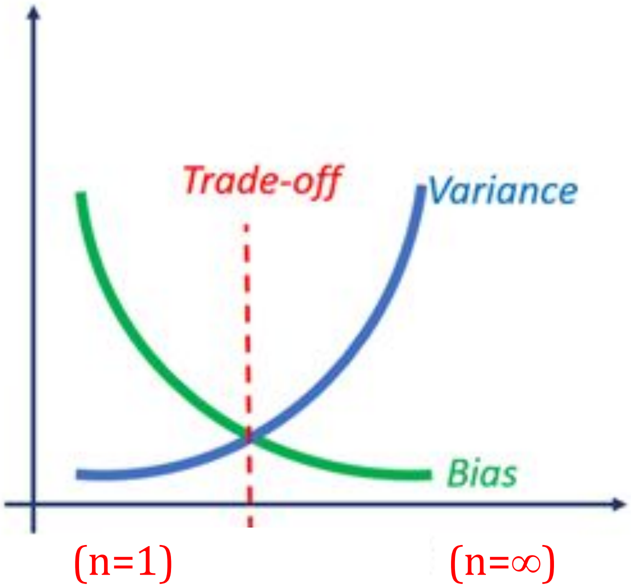
$$n > 1$$

Why bootstrap?

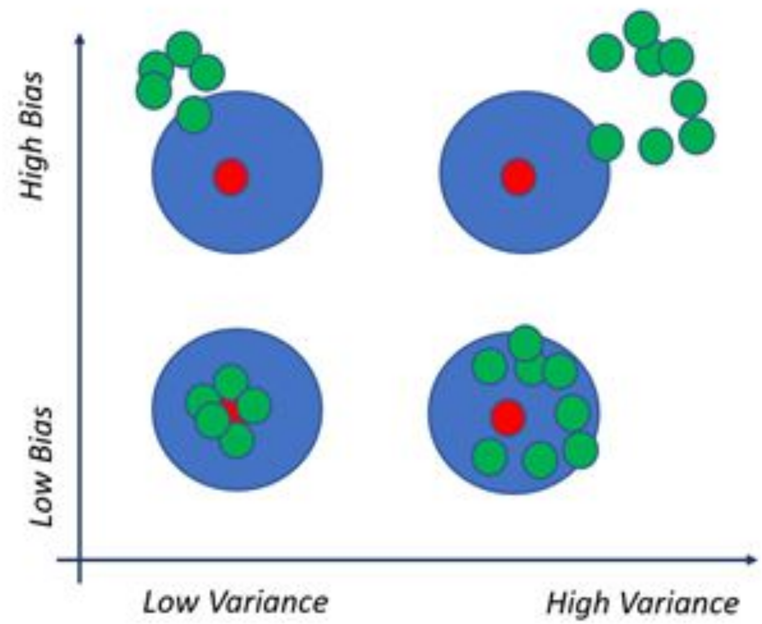


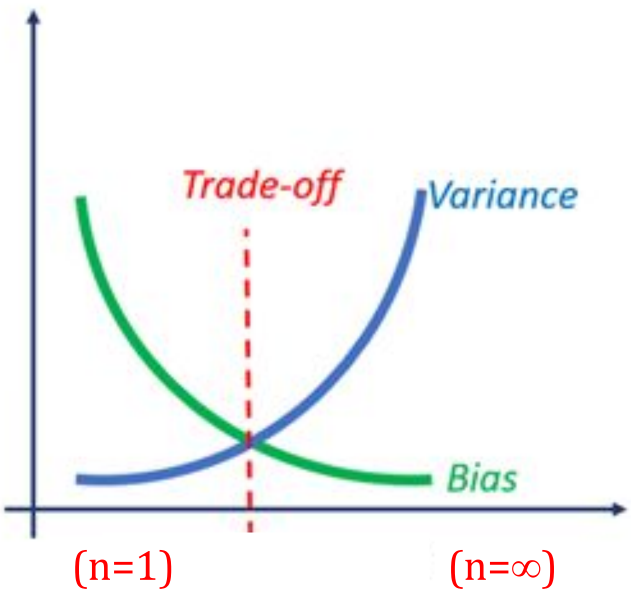
Why bootstrap?





Why bootstrap?





Bias-variance trade-off for the return estimate depth (n)

Actor-critic policy gradient with bootstrap

1. Collect trace

Actor-critic policy gradient with bootstrap

1. Collect trace

2. Estimate cumulative return for each step in trace

$$\hat{Q}_{\mathrm{n}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\phi(s_{t+n})$$

Actor-critic policy gradient with bootstrap

1. Collect trace

2. Estimate cumulative return for each step in trace

$$\hat{Q}_{\mathrm{n}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\phi(s_{t+n})$$

3. Train value network, e.g., on squared loss

$$L(\phi | s_t, a_t) = \left(\hat{Q}_n(s_t, a_t) - V_\phi(s_t)\right)^2$$

Actor-critic policy gradient with bootstrap

1. Collect trace

2. Estimate cumulative return for each step in trace

$$\hat{Q}_{\mathrm{n}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\phi(s_{t+n})$$

3. Train value network, e.g., on squared loss

$$L(\phi | s_t, a_t) = \left(\hat{Q}_n(s_t, a_t) - V_\phi(s_t)\right)^2$$

4. Train policy with policy gradient

$$\nabla_\theta L(\theta | s_t, a_t) = \hat{Q}_n(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta(a_t | s_t)]$$

Actor-critic policy gradient with bootstrap

Algorithm 4: Actor-critic policy gradient with bootstrapping

Input: A policy $\pi_\theta(a|s)$, a value function $V_\phi(s)$
A estimation depth n, learning rate η, number of episode M.
Initialization: Randomly initialize θ and ϕ.
while *not converged* **do**
    grad $\leftarrow 0$
    for $i = 1, .., M$ **do**
        Sample trace $h_0 = \{s_0, a_0, r_0, s_1, .., s_{T+1}\}$ following $\pi_\theta(a|s)$
        for $t = 0..T$ **do**
            $\hat{Q}_n(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\theta(s_{t+n})$ `/* n-step target */`
        end
    end
    $\phi \leftarrow \phi - \eta \cdot \nabla_\phi \sum_t (\hat{Q}_n(s_t, a_t) - V_\phi(s_t))^2$ `/* Update value */`
    $\theta \leftarrow \theta + \eta \cdot \sum_t [\hat{Q}_n(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)]$ `/* Update policy */`
end
Return $\pi_\theta(a|s)$

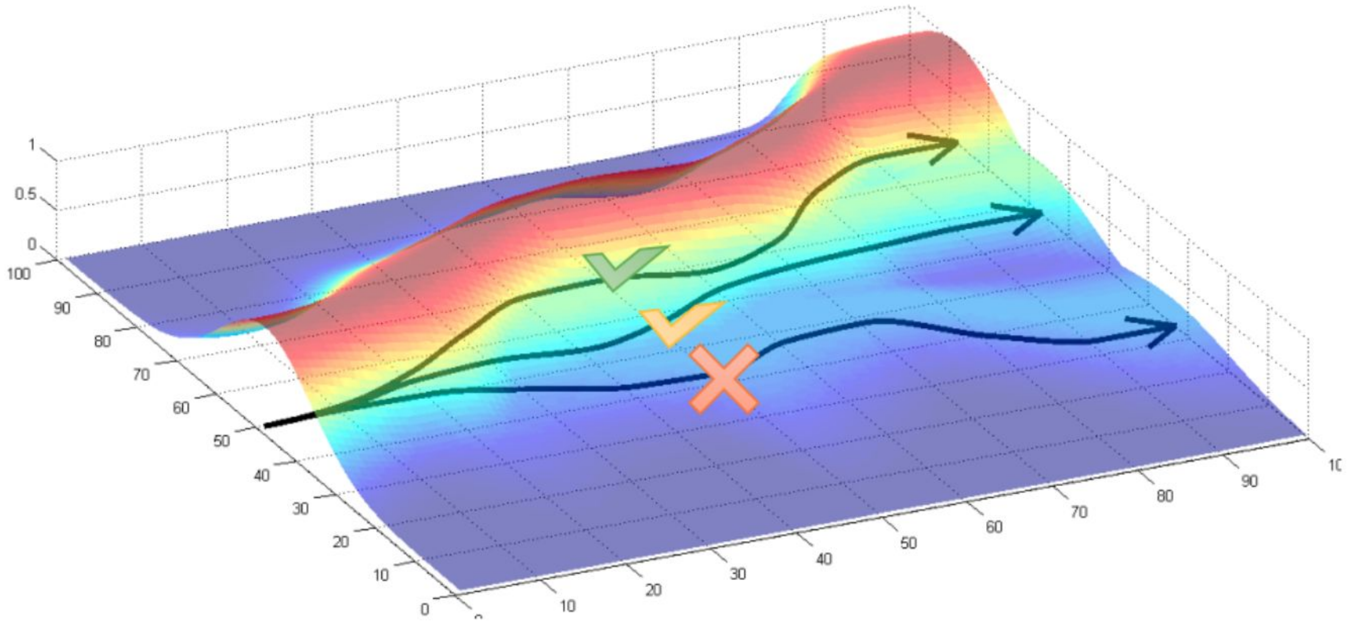
Full algorithm in lecture notes

Actor-critic

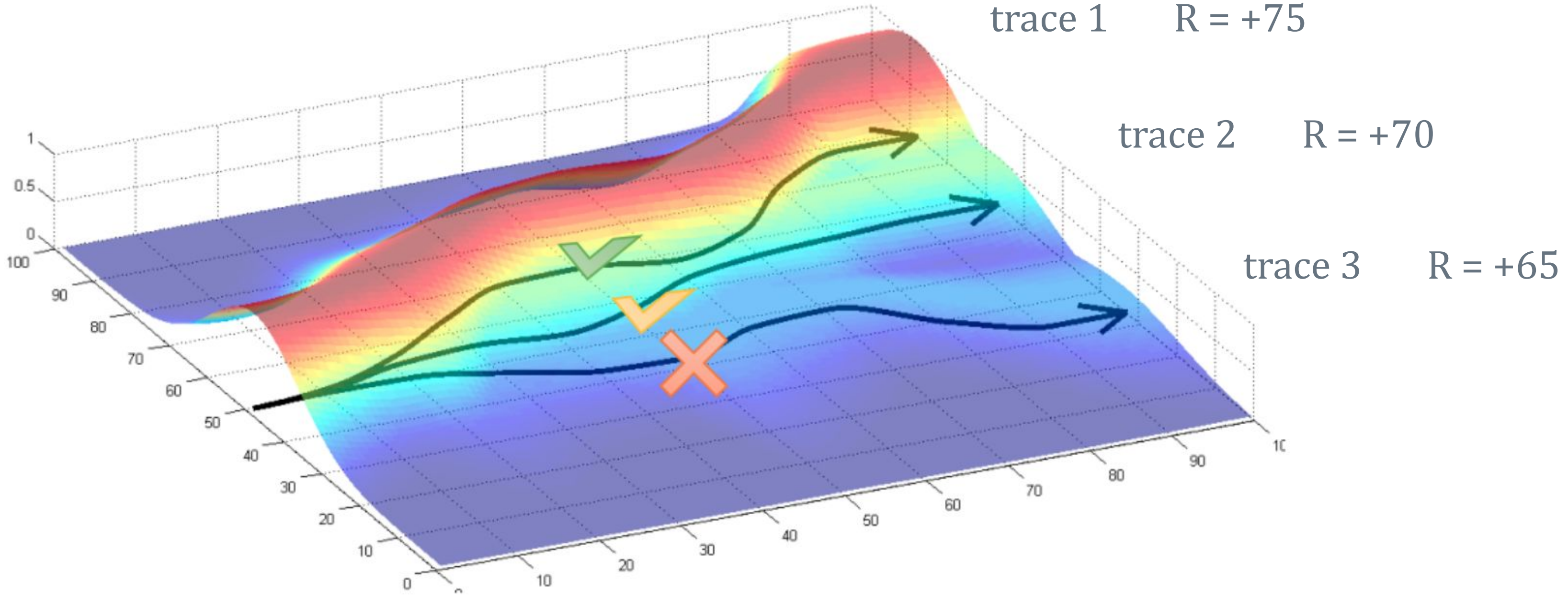
Use a value function to potentially get a better policy gradient update

1. Within policy gradient theorem:
   a. Bootstrapping — lower variance in cumulative reward estimate
   b. Baseline subtraction — lower variance in gradient estimate
2. Other type of update:
   a. Deterministic policy gradient

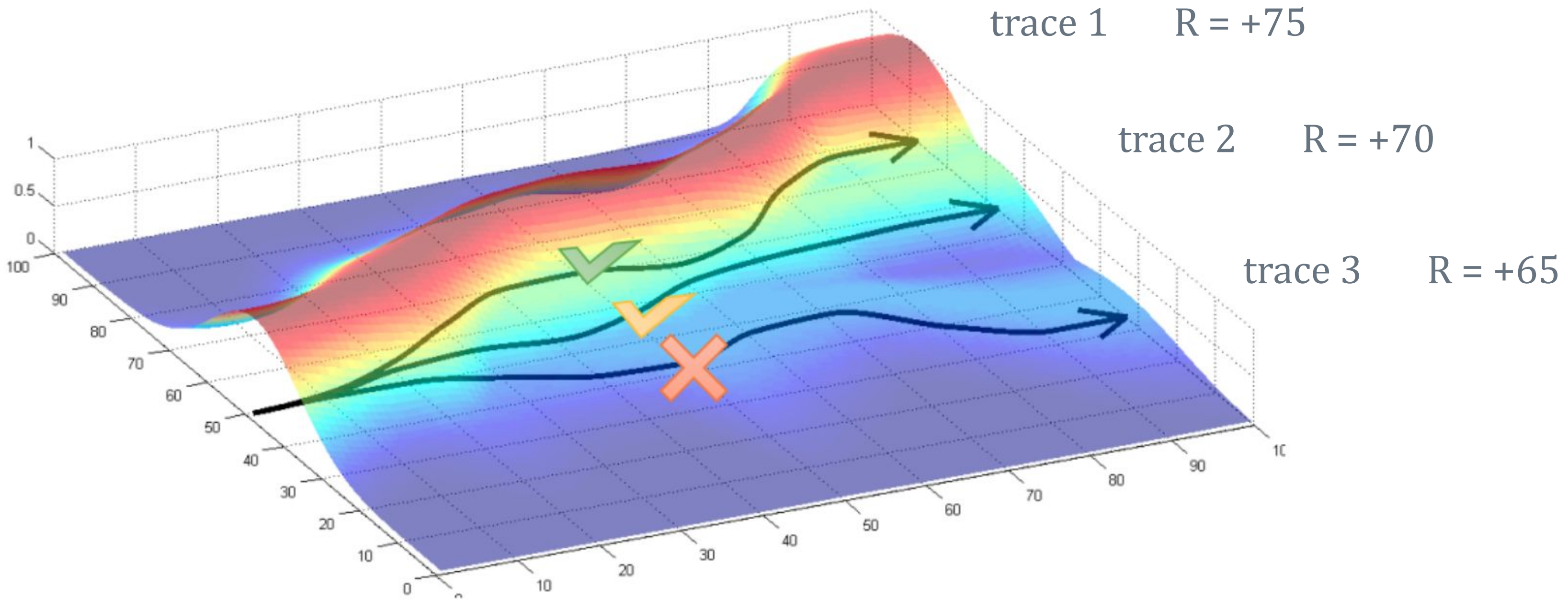
Baseline subtraction



Baseline subtraction

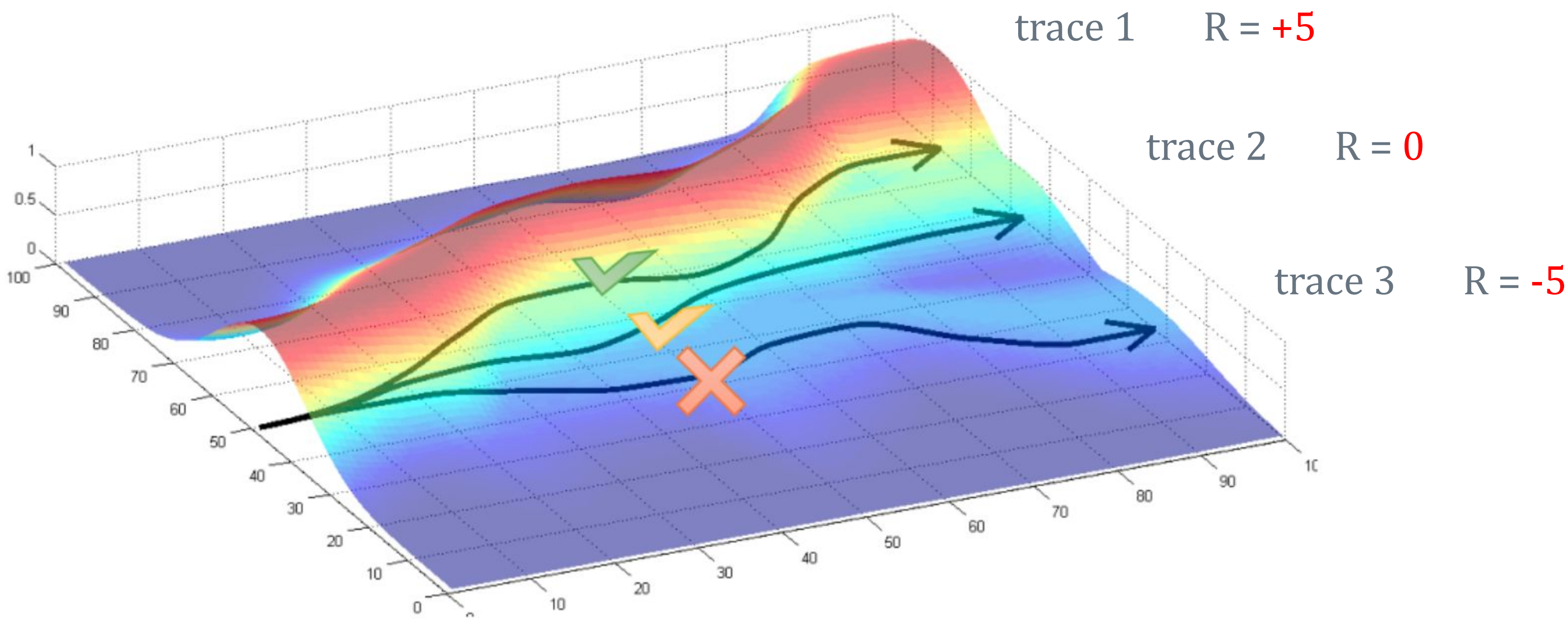


Baseline subtraction

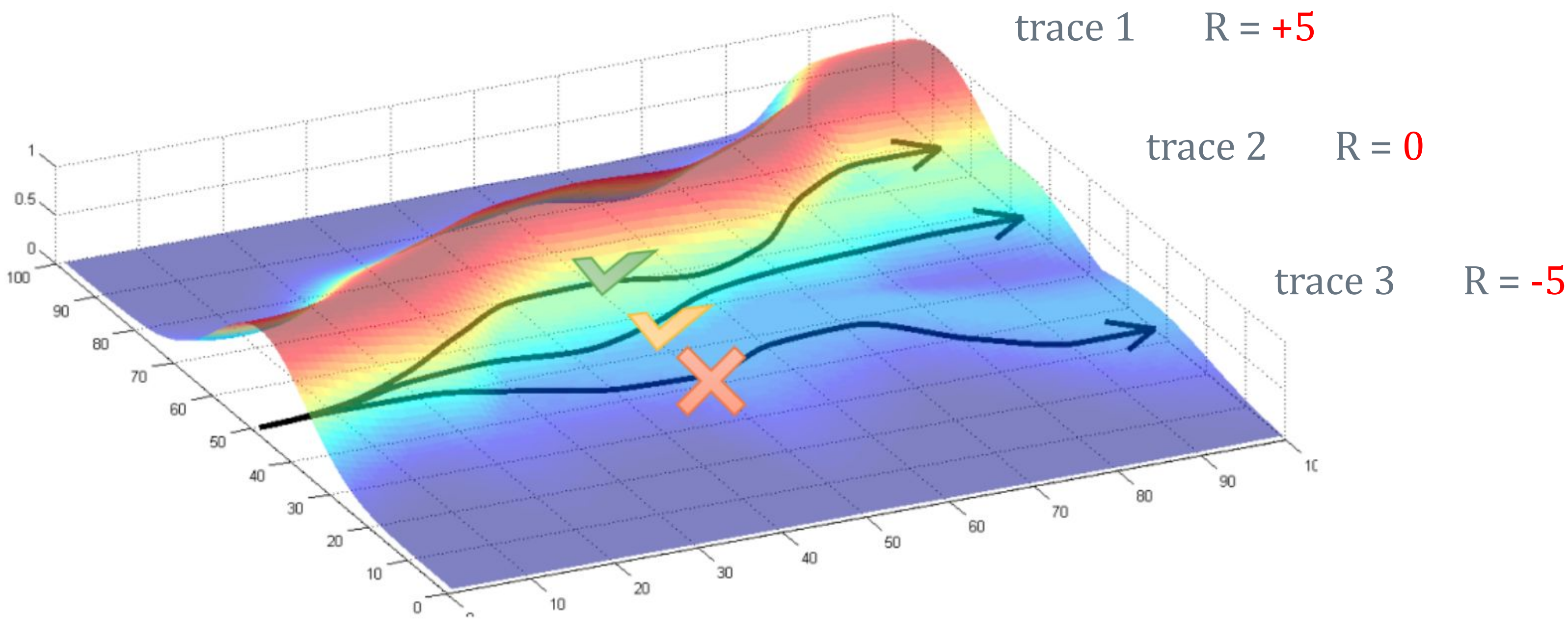


Probability of all actions will be pushed up, just trace 1 gets pushed harder
(when we sample 3 without 1, then 3 will still go up)

Baseline subtraction



Baseline subtraction



Now it would work much better!

Advantage function

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

Advantage function

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

Most common baseline is state value:

gives the "advantage", i.e.,

"how much better is an action than the state average"

Actor-critic PG with bootstrap & baseline

1. Collect trace

2. Estimate cumulative return for each step in trace

$$\hat{Q}_{\mathrm{n}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\phi(s_{t+n})$$

3. Train value network, e.g., on squared loss

$$L(\phi|s_t, a_t) = \left(\hat{Q}_n(s_t, a_t) - V_\phi(s_t)\right)^2$$

4. Train policy with policy gradient

$$\nabla_\theta L(\theta|s_t, a_t) = \hat{A}_n(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)]$$ with $$\hat{A}_{\mathrm{n}}(s_t, a_t) = \hat{Q}_n(s_t, a_t) - V_\phi(s_t)$$

Actor-critic PG with bootstrap & baseline

1. Collect trace

2. Estimate cumulative return for each step in trace

$$\hat{Q}_{\mathrm{n}}(s_t, a_t) = \sum_{k=0}^{n-1} r_{t+k} + V_\phi(s_{t+n})$$

3. Train value network, e.g., on squared loss

$$L(\phi | s_t, a_t) = \left(\hat{Q}_n(s_t, a_t) - V_\phi(s_t)\right)^2$$

4. Train policy with policy gradient

$$\nabla_\theta L(\theta | s_t, a_t) = \boxed{\hat{A}_n(s_t, a_t)} \cdot \nabla_\theta \log \pi_\theta(a_t | s_t)]$$

with $\boxed{\hat{A}_{\mathrm{n}}(s_t, a_t) = \hat{Q}_n(s_t, a_t) - V_\phi(s_t)}$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\left[\sum_{t=0}^{n} \Psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)} \left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)} \left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

$$\Psi_t = \hat{Q}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i$$

Monte Carlo target

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)} \left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

$$\Psi_t = \hat{Q}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i \qquad \text{Monte Carlo target}$$

$$\Psi_t = \hat{Q}_n(s_t, a_t) \quad = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) \qquad \text{bootstrap (n-step target)}$$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)}\left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

$$\Psi_t = \hat{Q}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i \qquad \text{Monte Carlo target}$$

$$\Psi_t = \hat{Q}_n(s_t, a_t) = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) \qquad \text{bootstrap (n-step target)}$$

$$\Psi_t = \hat{A}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i - V_\theta(s_t) \qquad \text{Baseline subtraction}$$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)} \left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

$$\Psi_t = \hat{Q}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i \qquad \text{Monte Carlo target}$$

$$\Psi_t = \hat{Q}_n(s_t, a_t) \quad = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) \qquad \text{bootstrap (n-step target)}$$

$$\Psi_t = \hat{A}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i - V_\theta(s_t) \qquad \text{Baseline subtraction}$$

$$\Psi_t = \hat{A}_n(s_t, a_t) \quad = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) - V_\theta(s_t) \qquad \text{Baseline + bootstrap}$$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)} \left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

$$\Psi_t = \hat{Q}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i \qquad \text{Monte Carlo target}$$

$$\Psi_t = \hat{Q}_n(s_t, a_t) \quad = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) \qquad \text{bootstrap } (n\text{-step target})$$

$$\Psi_t = \hat{A}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i - V_\theta(s_t) \qquad \text{Baseline subtraction}$$

$$\Psi_t = \hat{A}_n(s_t, a_t) \quad = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) - V_\theta(s_t) \qquad \text{Baseline + bootstrap}$$

$$\Psi_t = Q_\theta(s_t, a_t) \qquad \text{Q-value approximation}$$

General policy gradient formulation

$$\nabla_\theta J(\theta) = \mathbb{E}_{h_0 \sim p_\theta(h_0)} \left[\sum_{t=0}^{n} \boxed{\Psi_t} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

$$\Psi_t = \hat{Q}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i \qquad \text{Monte Carlo target}$$

$$\Psi_t = \hat{Q}_n(s_t, a_t) = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) \qquad \text{bootstrap (n-step target)}$$

$$\Psi_t = \hat{A}_{MC}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^i \cdot r_i - V_\theta(s_t) \qquad \text{Baseline subtraction}$$

$$\Psi_t = \hat{A}_n(s_t, a_t) = \sum_{i=t}^{n-1} \gamma^i \cdot r_i + \gamma^n V_\theta(s_n) - V_\theta(s_t) \qquad \text{Baseline + bootstrap}$$

$$\Psi_t = Q_\theta(s_t, a_t) \qquad \text{Q-value approximation}$$

These principles apply to all reinforcement learning (also value-based)!

Actor-critic

Use a value function to potentially get a better policy gradient update

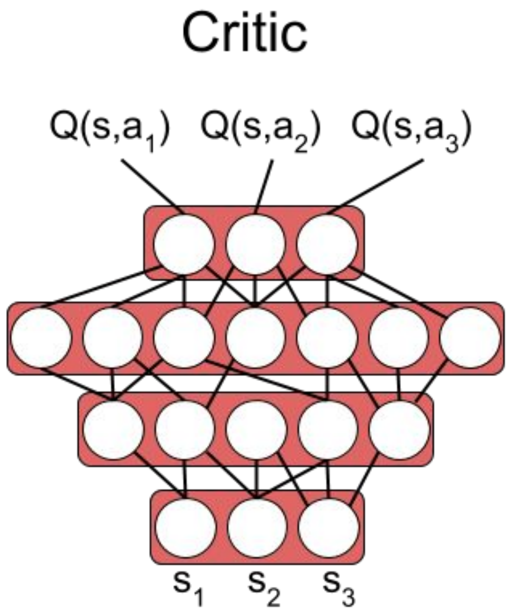
1. Within policy gradient theorem:
    a. Bootstrapping — lower variance in cumulative reward estimate
    b. Baseline subtraction — lower variance in gradient estimate
2. Other type of update:
    a. Deterministic policy gradient

Deterministic policy gradient

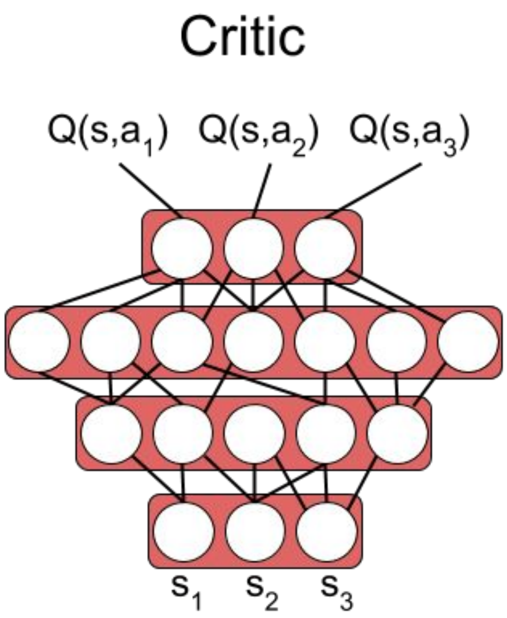
Deterministic policy gradient

- Alternative policy gradient approach
- Only conceptually discussed (short)

Deterministic policy gradient



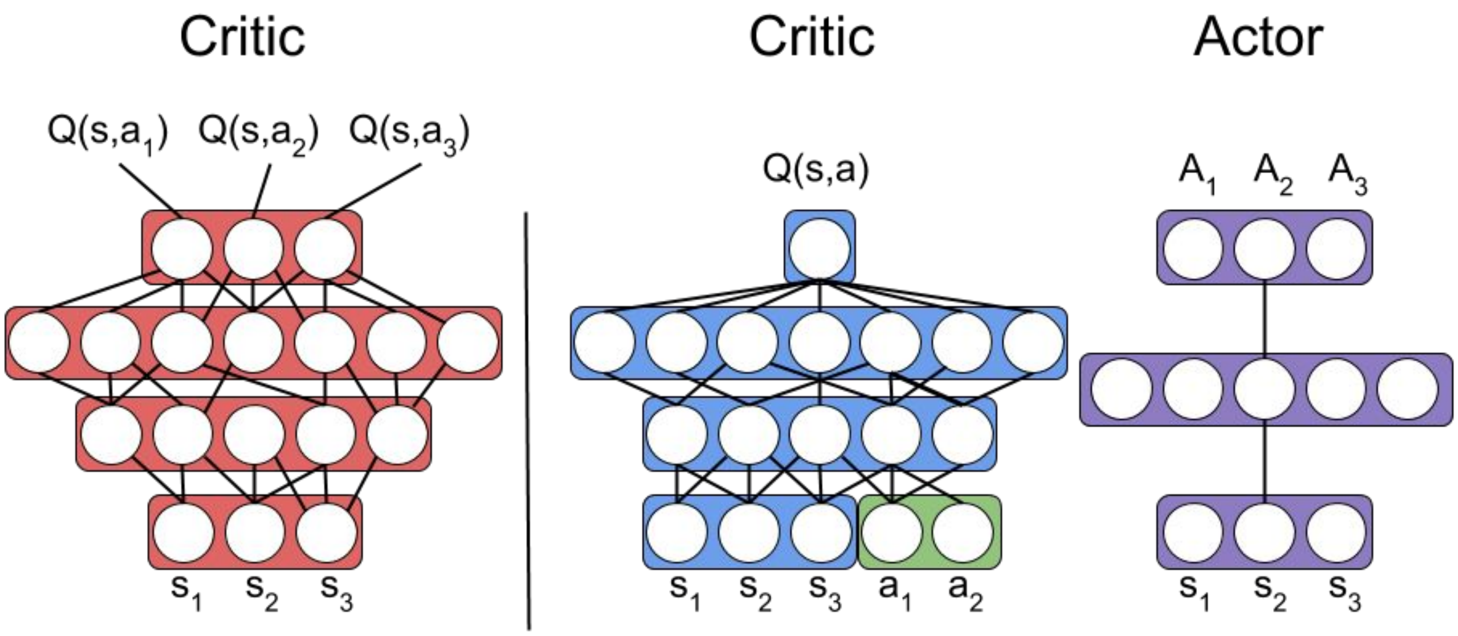
Deterministic policy gradient



Value-based RL

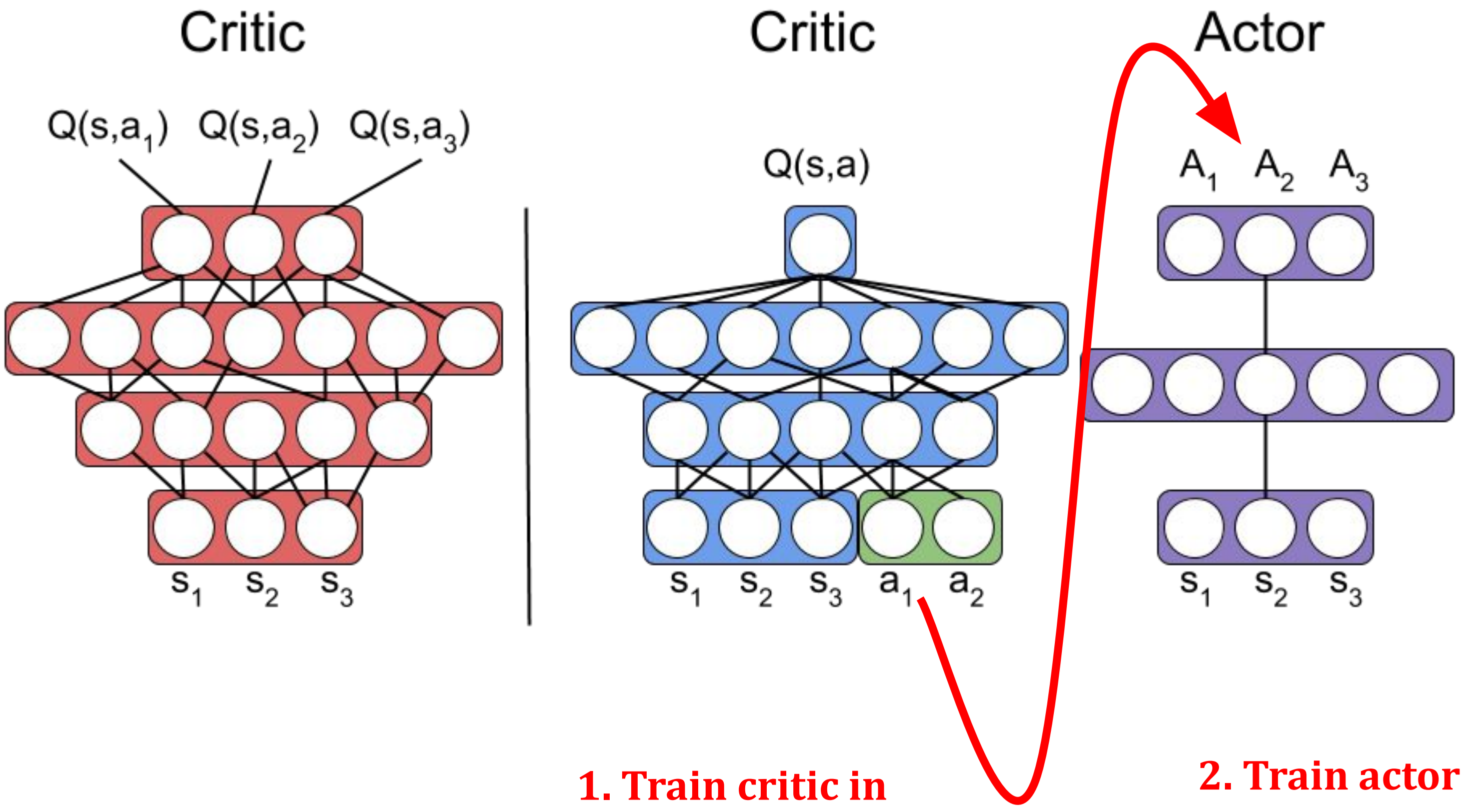
E.g. deep Q-network (DQN)

Deterministic policy gradient



1. **Train critic in same way**

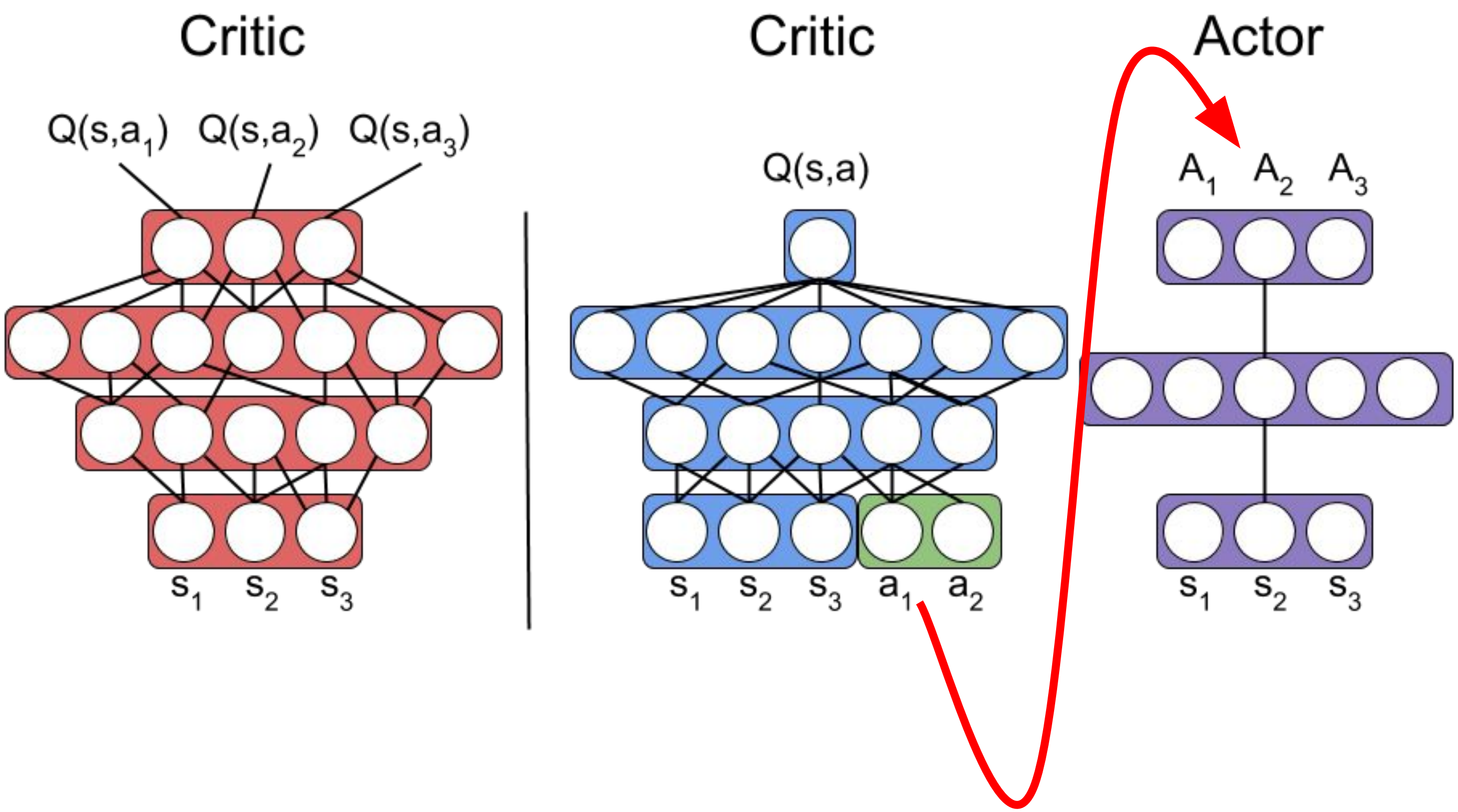
Deterministic policy gradient



1. Train critic in same way

2. Train actor by pushing gradient through the actions

Deterministic policy gradient



= Deterministic policy gradient
(only need to understand concept)

6. Gradient-free policy search

Gradient-free optimization

Can also do gradient-free optimization:

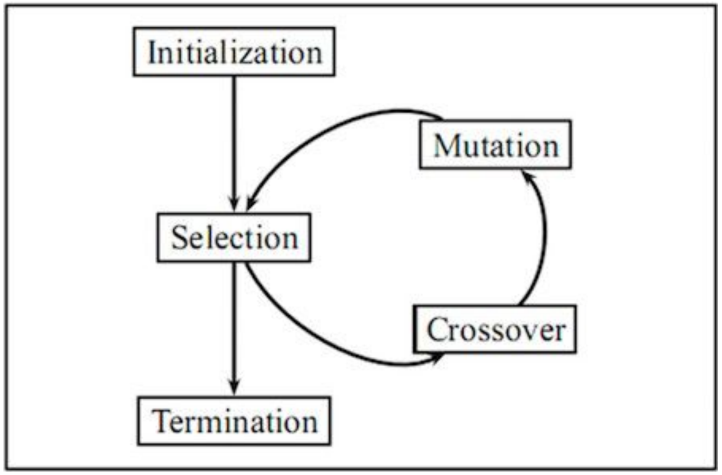
- Evolutionary strategies
- Cross-entropy method
- Simulated annealing

Gradient-free optimization

Can also do gradient-free optimization:

- Evolutionary strategies
- Cross-entropy method
- Simulated annealing

Population-based



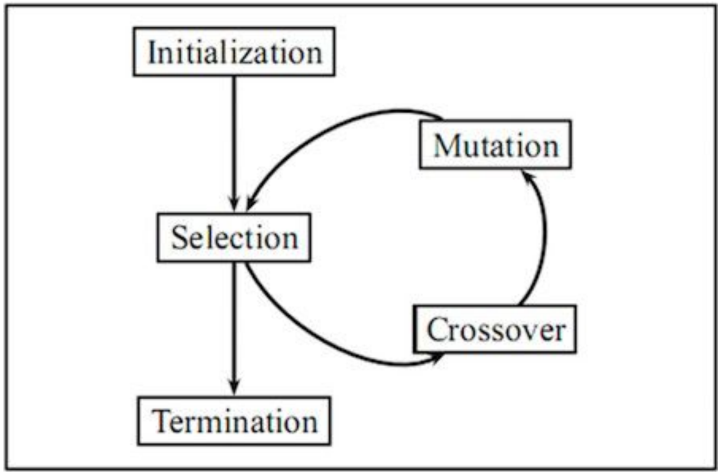
Gradient-free optimization

Can also do gradient-free optimization:

- Evolutionary strategies
- Cross-entropy method
- Simulated annealing

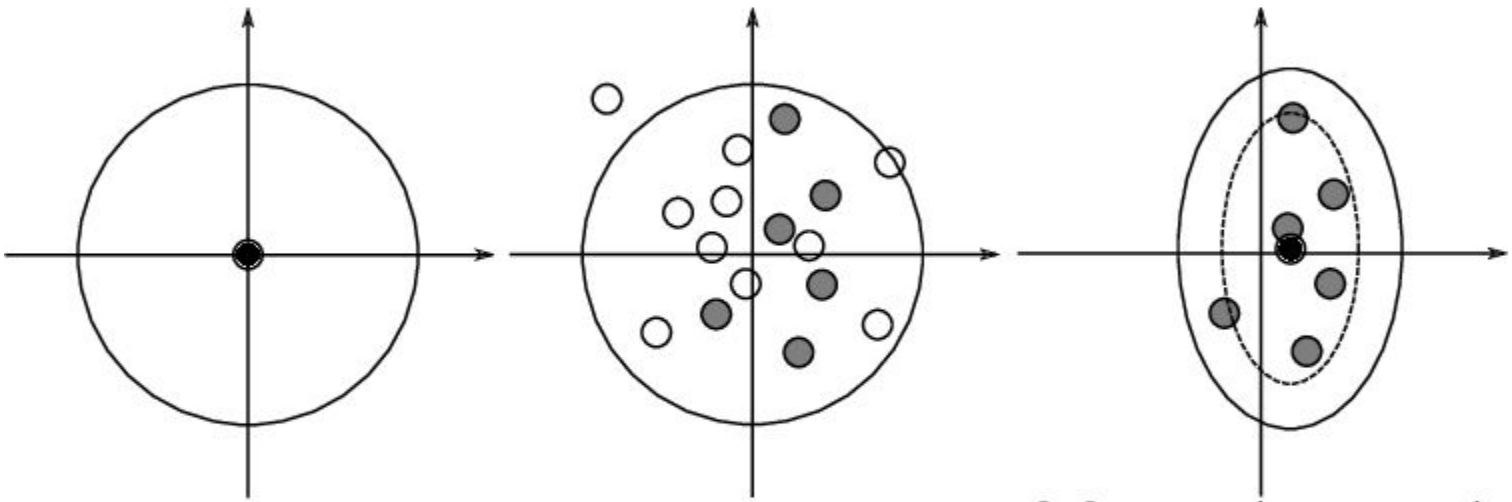
Population-based

Received less attention in RL and ML compared to gradient-based optimization, but start to resurface!



Cross-entropy method

Cross-entropy method

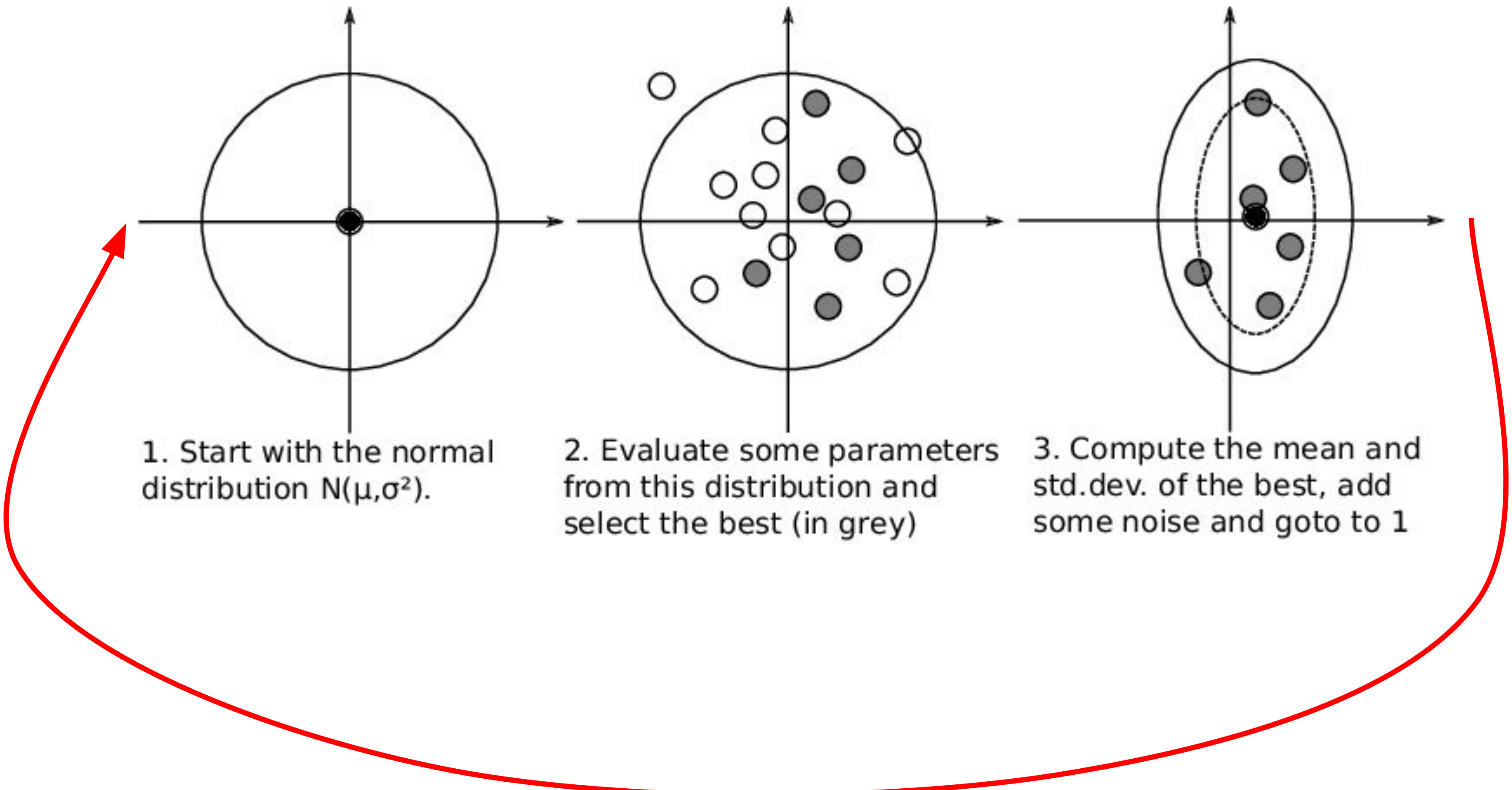


1. Start with the normal distribution N(μ,σ²).

2. Evaluate some parameters from this distribution and select the best (in grey)

3. Compute the mean and std.dev. of the best, add some noise and goto to 1

Cross-entropy method



Cross-entropy method

Very simple policy search method & often a strong baseline

Cross-entropy method

Very simple policy search method & often a strong baseline

Algorithm 4: Cross-entropy method (CEM) for reinforcement learning

Input: A differentiable policy $\pi_\theta(a|s)$, parametrized by $\theta \in \mathbb{R}^d$, where $\theta \sim \mathcal{N}(\mu, \sigma)$. Number of iterations n_{iter}, number of samples n_{sample}, top percentage u.

Initialization: Randomly initialize $\mu_1 \in \mathbb{R}^d$ and $\sigma_1 \in (\mathbb{R}^+)d$.

for $i = 1..n_{iter}$ **do**

- Sample parameters $\theta_j \sim \mathcal{N}(\mu_i, \text{diag}(\sigma_i))$ for $j = 1..n_{\text{sample}}$
- Sample returns $R_j \sim \pi_{\theta_j}(a|s)$
- Select elite set of θ_j giving the top $u\%$ returns R_j
- $\mu_{i+1}, \sigma_{i+1} \leftarrow$ refit Gaussian on elite set

end

$\theta = \mu$

Return $\pi_\theta(a|s)$

Cross-entropy method

Very simple policy search method & often a strong baseline

Algorithm 4: Cross-entropy method (CEM) for reinforcement learning

---

Input: A differentiable policy $\pi_\theta(a|s)$, parametrized by $\theta \in \mathbb{R}^d$, where $\theta \sim \mathcal{N}(\mu, \sigma)$. Number of iterations n_{iter}, number of samples n_{sample}, top percentage u.

Initialization: Randomly initialize $\mu_1 \in \mathbb{R}^d$ and $\sigma_1 \in (\mathbb{R}^+)d$.

for $i = 1..n_{iter}$ **do**

  Sample parameters $\theta_j \sim \mathcal{N}(\mu_i, \text{diag}(\sigma_i))$ for $j = 1..n_{\text{sample}}$

  Sample returns $R_j \sim \pi_{\theta_j}(a|s)$

  Select elite set of θ_j giving the top $u\%$ returns R_j

  $\mu_{i+1}, \sigma_{i+1} \leftarrow$ refit Gaussian on elite set

end

$\theta = \mu$

Return $\pi_\theta(a|s)$

Pseudo-code in lecture notes

Summary & Assignments

Summary

First half:

Summary

First half:

- Continuous RL is important for the real world!

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.
- It does alter the way we should specify our solution, with focus on policy-based and actor critic approaches.

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.
- It does alter the way we should specify our solution, with focus on policy-based and actor critic approaches.

Second half:

- Policy-based approaches can be gradient-based (policy gradients) or gradient-free.

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.
- It does alter the way we should specify our solution, with focus on policy-based and actor critic approaches.

Second half:

- Policy-based approaches can be gradient-based (policy gradients) or gradient-free.
- Gradient-based:
    - The cardinal policy gradient algorithm use the *policy-gradient theorem.*

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.
- It does alter the way we should specify our solution, with focus on policy-based and actor critic approaches.

Second half:

- Policy-based approaches can be gradient-based (policy gradients) or gradient-free.
- Gradient-based:
  - The cardinal policy gradient algorithm use the *policy-gradient theorem.*
  - This also has various actor-critic extensions.

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.
- It does alter the way we should specify our solution, with focus on policy-based and actor critic approaches.

Second half:

- Policy-based approaches can be gradient-based (policy gradients) or gradient-free.
- Gradient-based:
    - The cardinal policy gradient algorithm use the *policy-gradient theorem.*
    - This also has various actor-critic extensions.
    - An alternative policy gradient approach are *deterministic policy gradients*.

Summary

First half:

- Continuous RL is important for the real world!
- It does not alter the MDP specification much.
- It does alter the way we should specify our solution, with focus on policy-based and actor critic approaches.

Second half:

- Policy-based approaches can be gradient-based (policy gradients) or gradient-free.
- Gradient-based:
    - The cardinal policy gradient algorithm use the *policy-gradient theorem.*
    - This also has various actor-critic extensions.
    - An alternative policy gradient approach are *deterministic policy gradients*.
- Gradient-free:
    - Cross-entropy method

To Do

- Read the Lecture Notes!
  - More mathematical notation, but you need to get familiar with this to understand, implement and describe algorithms.
  - Focus on the sections “Representation of Solution” and entire Part “Policy-based RL”

To Do

- Read the Lecture Notes!
  - More mathematical notation, but you need to get familiar with this to understand, implement and describe algorithms.
  - Focus on the sections “Representation of Solution” and entire Part “Policy-based RL”

- Make the assignment!
  - Most practical emphasis on vanilla policy gradient (REINFORCE)

To Do

- Read the Lecture Notes!
  - More mathematical notation, but you need to get familiar with this to understand, implement and describe algorithms.
  - Focus on the sections “Representation of Solution” and entire Part “Policy-based RL”
- Make the assignment!
  - Most practical emphasis on vanilla policy gradient (REINFORCE)