

# Reinforcement Learning

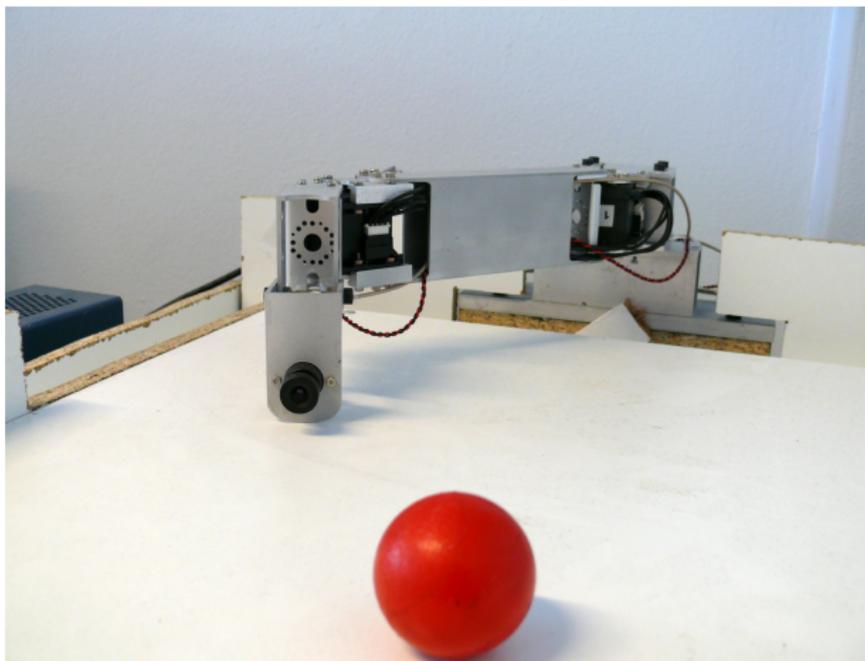
## Part I: The Classical Setting

*Ivan Koryakovskiy*   Jens Kober   Ivo Grondman  
Robert Babuška

Knowledge-Based Control Systems

# Demo: RL for a robot goalkeeper

Learn how to catch ball, using video camera image



# Outline

- 1 Learning paradigms
- 2 Elements of RL
- 3 Algorithms
- 4 Summary and outlook

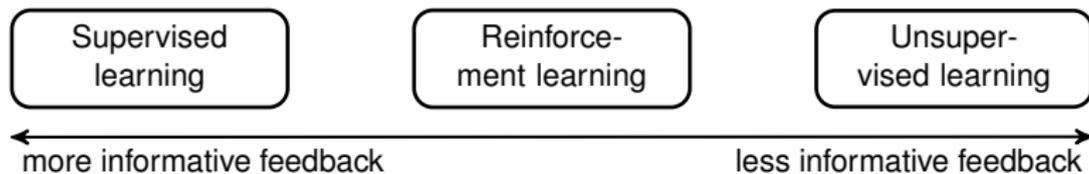
# Why learning?

**Learning** can find solutions that:

- 1 cannot be found in advance
  - problem too complex  
(e.g., controlling highly nonlinear systems)
  - problem not fully known beforehand  
(e.g., robotic exploration of extraterrestrial planets)
- 2 steadily improve
- 3 adapt to time-varying environments

Essential for any **intelligent** system

# RL on the Machine Learning spectrum



# Spectrum: Supervised learning

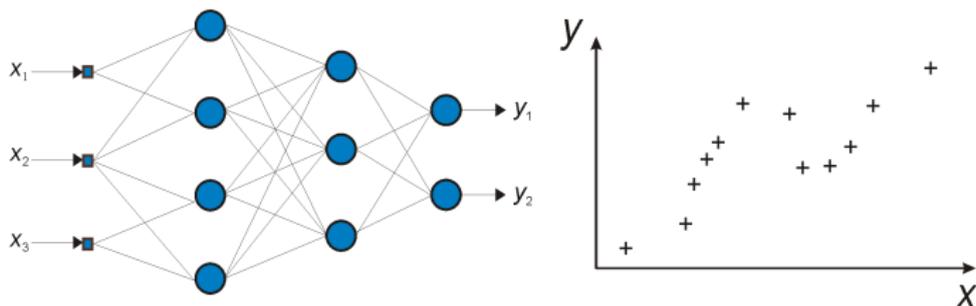


- For each input sample  $x$ , **correct output**  $y$  is known
- Infer input-output relationship  $y \approx g(x)$

# Spectrum: Supervised learning



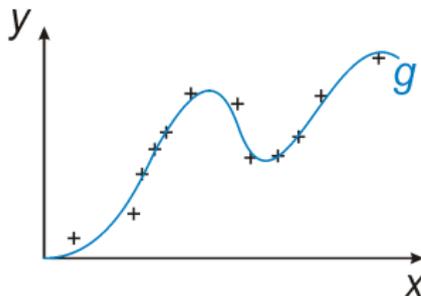
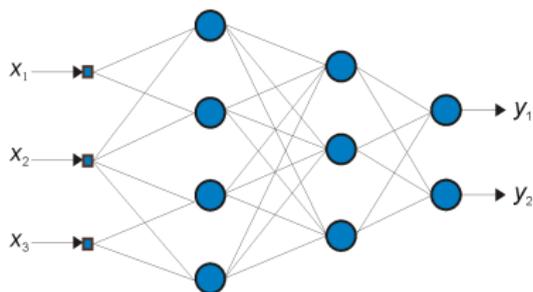
- For each input sample  $x$ , **correct output**  $y$  is known
- Infer input-output relationship  $y \approx g(x)$
- Example: **neural networks**



# Spectrum: Supervised learning



- For each input sample  $x$ , **correct output**  $y$  is known
- Infer input-output relationship  $y \approx g(x)$
- Example: **neural networks**



# Spectrum: Unsupervised learning



- Only input samples available – **no outputs**
- Find patterns in the data

# Spectrum: Unsupervised learning



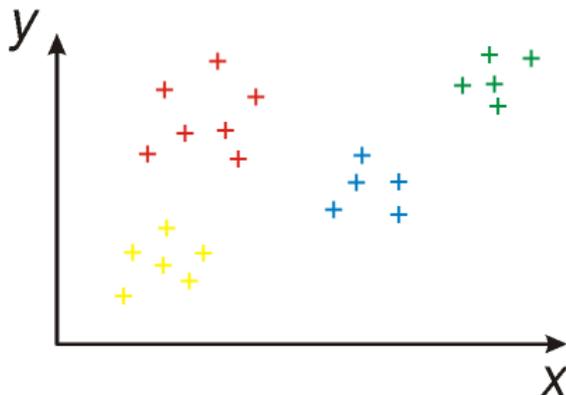
- Only input samples available – **no outputs**
- Find patterns in the data
- Example: **clustering**



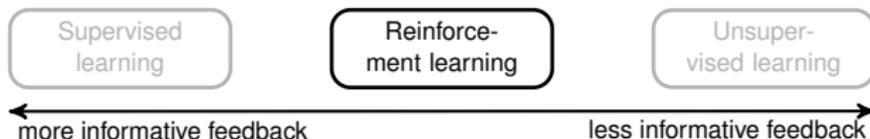
# Spectrum: Unsupervised learning



- Only input samples available – **no outputs**
- Find patterns in the data
- Example: **clustering**



# Spectrum: Reinforcement learning



- Correct outputs not available, **only rewards**
- Find optimal control behavior

1 Learning paradigms

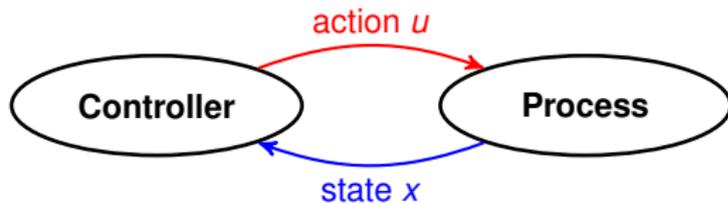
2 Elements of RL

Markov decision process, learning goal, policy  
Bellman equation, optimality, solutions

3 Algorithms

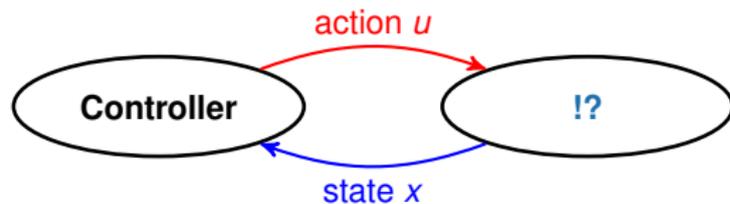
4 Summary and outlook

# Principle of RL



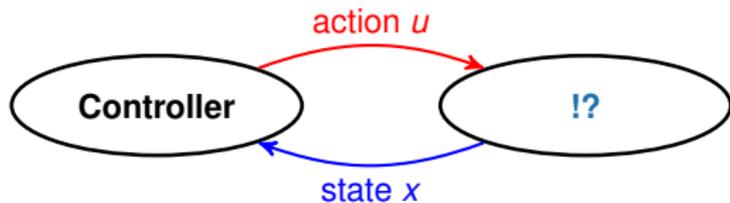
- Interact with a system through **states** and **actions**

# Principle of RL



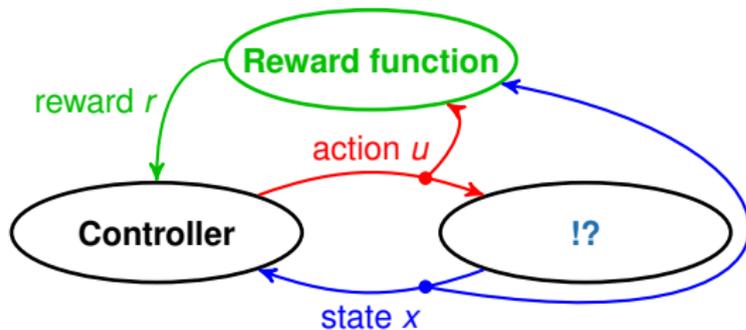
- Interact with a system through **states** and **actions**

# Principle of RL



- Interact with a system through **states** and **actions**
- Inspired by human and animal learning

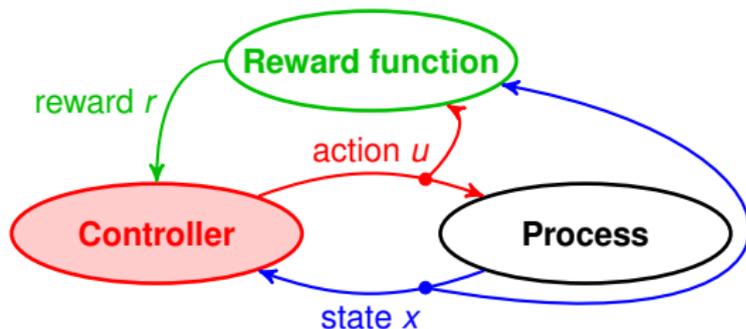
# Principle of RL



- Interact with a system through **states** and **actions**
- Inspired by human and animal learning
- Receive **rewards** as performance feedback

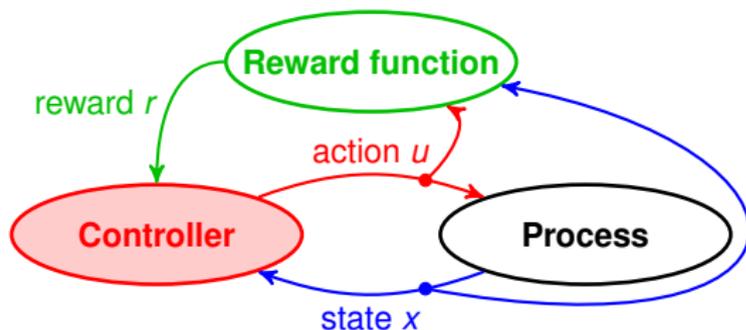
# Reinforcement learning = Control

Reinforcement learning is about **control**:  
optimal, adaptive, and model-free



# Reinforcement learning = Control

Reinforcement learning is about **control**:  
optimal, adaptive, and model-free



This lecture: **classical RL** – discrete states and actions

1 Learning paradigms

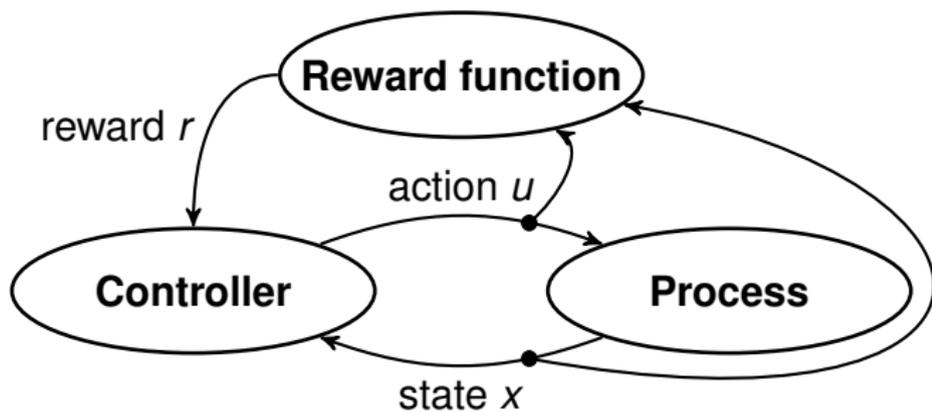
2 Elements of RL

Markov decision process, learning goal, policy  
Bellman equation, optimality, solutions

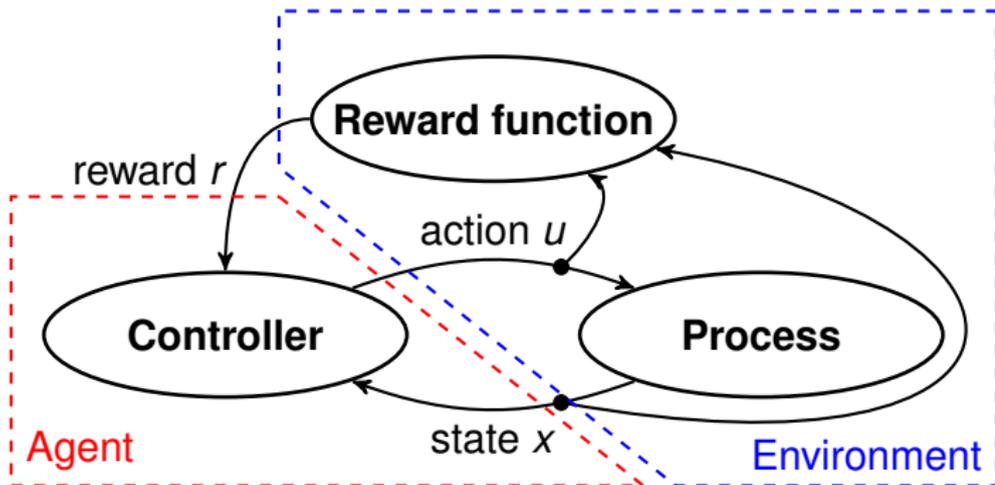
3 Algorithms

4 Summary and outlook

# Environment and agent



# Environment and agent



# The environment

The environment is modeled by an MDP:

## Markov Decision Process (MDP)

An MDP is a tuple  $\langle X, U, f, \rho \rangle$  where:

- $X$  is the finite state space
- $U$  is the finite action space
- $f : X \times U \rightarrow X$  is the state transition function
- $\rho : X \times U \rightarrow \mathbb{R}$  is the reward function

$x_{k+1} = f(x_k, u_k)$ , with  $k$  the discrete time

$r_k = \rho(x_k, u_k)$

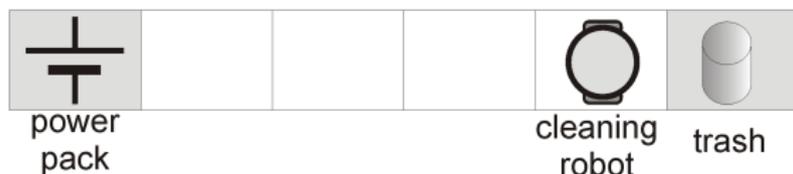
Note: stochastic formulation is possible

# The agent

The agent is a state feedback controller:

- Learns optimal mapping from states to actions
- **Policy**  $\pi : X \mapsto U$  is the control law

# A simple cleaning robot example



- Cleaning robot in a 1-D world
- Goal: pick up trash (reward +5) or power pack (reward +1)
- After picking up item, episode terminates

## Cleaning robot: State & action



- Robot in given state  $x$  (cell)
- and takes action  $u$  (e.g., move right)

## Cleaning robot: State & action



- Robot in given state  $x$  (cell)
- and takes action  $u$  (e.g., move right)



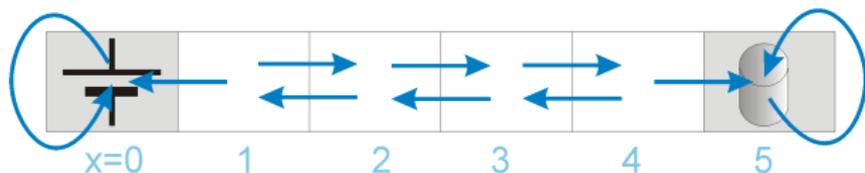
- State space  $X = \{0, 1, 2, 3, 4, 5\}$
- Action space  $U = \{-1, 1\} = \{\text{left}, \text{right}\}$

# Cleaning robot: Transition & reward



- Robot reaches **next state**  $x'$
- and receives **reward**  $r$  = quality of transition (here, +5 for collecting trash)

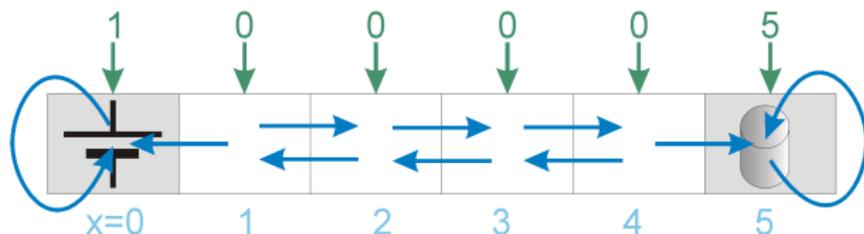
# Cleaning robot: Transition & reward functions



- **Transition function** (process behavior):

$$x' = f(x, u) = \begin{cases} x & \text{if } x \text{ is terminal (0 or 5)} \\ x + u & \text{otherwise} \end{cases}$$

# Cleaning robot: Transition & reward functions



- **Transition function** (process behavior):

$$x' = f(x, u) = \begin{cases} x & \text{if } x \text{ is terminal (0 or 5)} \\ x + u & \text{otherwise} \end{cases}$$

- **Reward function** (immediate performance):

$$r = \rho(x, u) = \begin{cases} 1 & \text{if } x = 1 \text{ and } u = -1 \text{ (powerpack)} \\ 5 & \text{if } x = 4 \text{ and } u = 1 \text{ (trash)} \\ 0 & \text{otherwise} \end{cases}$$

# Cleaning robot: Policy

- **Policy**  $\pi$ : mapping from  $x$  to  $u$  (state feedback)
- Determines controller behavior

# Cleaning robot: Policy

- **Policy**  $\pi$ : mapping from  $x$  to  $u$  (state feedback)
- Determines controller behavior

Example:



$$\pi(0) = *$$

$$\pi(1) = -1$$

$$\pi(2) = 1$$

$$\pi(3) = 1$$

$$\pi(4) = 1$$

$$\pi(5) = *$$

\* action irrelevant in terminal state

# Learning goal

Find  $\pi$  that maximizes **discounted return**:

$$R^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k))$$

from any  $x_0$

# Learning goal

Find  $\pi$  that maximizes **discounted return**:

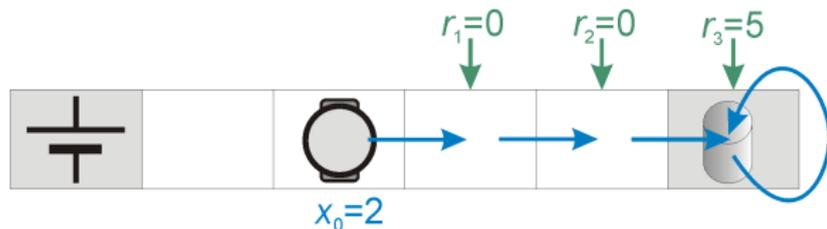
$$R^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k))$$

from any  $x_0$

Discount factor  $\gamma \in [0, 1)$ :

- induces a “pseudo-horizon” for optimization
- bounds infinite sum
- encodes increasing uncertainty about the future
- helps convergence of algorithms

## Cleaning robot: Return



Assume  $\pi$  always goes right

$$\begin{aligned} R^\pi(2) &= \gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \gamma^3 0 + \gamma^4 0 + \dots \\ &= \gamma^2 \cdot 5 \end{aligned}$$

Because  $x_3$  is terminal, all remaining rewards are 0

1 Learning paradigms

2 Elements of RL

Markov decision process, learning goal, policy  
**Bellman equation, optimality, solutions**

3 Algorithms

4 Summary and outlook

# Value function

One of these two is used:

- **V-function** (state value) of policy  $\pi$ :

$$V^\pi(x_0) = R^\pi(x_0)$$

- **Q-function** (state-action value) of policy  $\pi$ :

$$Q^\pi(x_0, u_0) = \rho(x_0, u_0) + \gamma R^\pi(x_1)$$

(return after taking  $u_0$  in  $x_0$  and then following  $\pi$ )

## Q-function

$$\begin{aligned}R^\pi(x_0) &= \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k)) \\&= \rho(x_0, \pi(x_0)) + \sum_{k=1}^{\infty} \gamma^k \rho(x_k, \pi(x_k)) \\&= \rho(x_0, \pi(x_0)) + \gamma \sum_{k=0}^{\infty} \gamma^k \rho(x_{k+1}, \pi(x_{k+1})) \\&= \rho(x_0, \pi(x_0)) + \gamma R^\pi(x_1)\end{aligned}$$

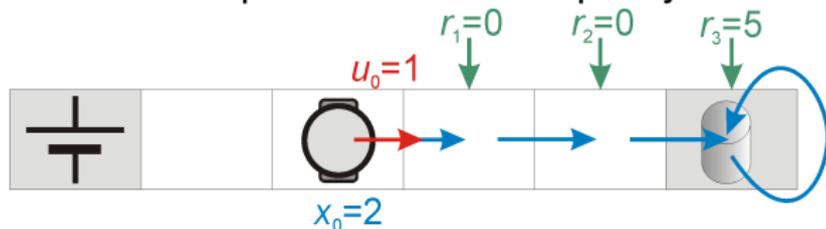
Q-function makes first action a free variable  $u_0$ :

$$Q^\pi(x_0, u_0) = \rho(x_0, u_0) + \gamma R^\pi(x_1)$$

## Q-function (cont'd)

$$Q^\pi(x_0, u_0) = \rho(x_0, u_0) + \gamma R^\pi(x_1)$$

- First action in the sequence independent of policy
- Rest of the sequence follows the policy



- Q-function allows direct derivation of policy

# Bellman equation

- Develop Q-function one step ahead:

$$\begin{aligned}Q^\pi(x_0, u_0) &= \rho(x_0, u_0) + \gamma R^\pi(x_1) \\ &= \rho(x_0, u_0) + \gamma[\rho(x_1, \pi(x_1)) + \gamma R^\pi(x_2)] \\ &= \rho(x_0, u_0) + \gamma Q^\pi(x_1, \pi(x_1))\end{aligned}$$

Remember:  $x_1 = f(x_0, u_0)$

# Bellman equation

- Develop Q-function one step ahead:

$$\begin{aligned}Q^\pi(x_0, u_0) &= \rho(x_0, u_0) + \gamma R^\pi(x_1) \\ &= \rho(x_0, u_0) + \gamma[\rho(x_1, \pi(x_1)) + \gamma R^\pi(x_2)] \\ &= \rho(x_0, u_0) + \gamma Q^\pi(x_1, \pi(x_1))\end{aligned}$$

Remember:  $x_1 = f(x_0, u_0)$

Bellman equation for  $Q^\pi$

$$Q^\pi(x, u) = \rho(x, u) + \gamma Q^\pi(f(x, u), \pi(f(x, u)))$$

# Optimal solution

- **Optimal Q-function:**

$$Q^* = \max_{\pi} Q^{\pi}$$

⇒ Greedy policy in  $Q^*$ :

$$\pi^*(x) = \arg \max_u Q^*(x, u)$$

is **optimal** (achieves maximal returns)

# Optimal solution

- **Optimal Q-function:**

$$Q^* = \max_{\pi} Q^{\pi}$$

⇒ Greedy policy in  $Q^*$ :

$$\pi^*(x) = \arg \max_u Q^*(x, u)$$

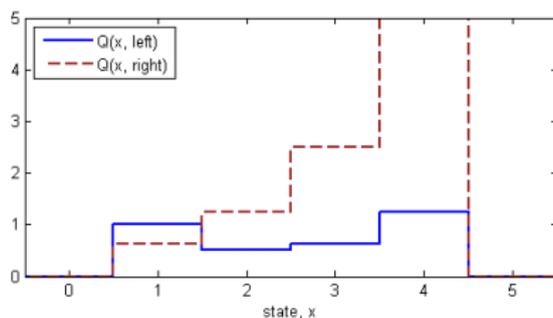
is **optimal** (achieves maximal returns)

Bellman optimality equation (for  $Q^*$ )

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(f(x, u), u')$$

# Cleaning robot: Optimal solution

Discount factor  $\gamma = 0.5$



1 Learning paradigms

2 Elements of RL

3 Algorithms

Taxonomy

Q-learning

SARSA

4 Summary and outlook

# Types of algorithms

## By model knowledge

- 1 **Model-based** – dynamic programming  
 $f, \rho$  known
- 2 **Model-free** – proper reinforcement learning  
 $f, \rho$  unknown, only transition data  $(x, u, x', r)$  available
- 3 **Model-learning** RL  
estimate  $f$  and  $\rho$  from transition data

# Types of algorithms (cont'd)

## By level of interaction

- 1 Offline  
data collected in advance
- 2 Online  
controller learns by interacting with the process

## By path to optimal solution

- 1 Off-policy  
find  $Q^*$ , use it to compute  $\pi^*$
- 2 On-policy  
find  $Q^\pi$ , improve  $\pi$ , repeat

# Algorithms in this lecture

**Online model-free** reinforcement learning:

Off-policy	On-policy
Q-learning	SARSA

Both methods are **temporal difference (TD)** methods

1 Learning paradigms

2 Elements of RL

3 Algorithms

Taxonomy

**Q-learning**

SARSA

4 Summary and outlook

# Off-policy online RL: Q-learning

Off-policy: **find**  $Q^*$ , use it to compute  $\pi^*$

- 1 Take Bellman optimality equation at some  $(x, u)$ :

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(f(x, u), u')$$

# Off-policy online RL: Q-learning

Off-policy: **find**  $Q^*$ , use it to compute  $\pi^*$

- 1 Take Bellman optimality equation at some  $(x, u)$ :

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(f(x, u), u')$$

- 2 Turn into **iterative update**:

$$Q(x, u) \leftarrow \rho(x, u) + \gamma \max_{u'} Q(f(x, u), u')$$

# Off-policy online RL: Q-learning

Off-policy: **find**  $Q^*$ , use it to compute  $\pi^*$

- 1 Take Bellman optimality equation at some  $(x, u)$ :

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u'} Q^*(f(x, u), u')$$

- 2 Turn into **iterative update**:

$$Q(x, u) \leftarrow \rho(x, u) + \gamma \max_{u'} Q(f(x, u), u')$$

- 3 Instead of model  $f, \rho$ , use **transition sample**

$(x_k, u_k, x_{k+1}, r_{k+1})$  at each step  $k$ :

$$Q(x_k, u_k) \leftarrow r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u')$$

Note:  $x_{k+1} = f(x_k, u_k)$ ,  $r_{k+1} = \rho(x_k, u_k)$

## Q-learning (cont'd)

- 4 Finally, make update **incremental**:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot [r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

with learning rate  $\alpha_k \in (0, 1]$ .

The expression

$$r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)$$

is called the **temporal difference**.

# Complete Q-learning algorithm

## Q-learning

**for** every trial **do**

initialize  $x_0$

**repeat** for each step  $k$

take action  $u_k$

measure  $x_{k+1}$ , receive  $r_{k+1}$

$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$

$$[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

**until** terminal state

**end for**

# Complete Q-learning algorithm

## Q-learning

**for** every trial **do**

initialize  $x_0$

**repeat** for each step  $k$

**take action**  $u_k$

measure  $x_{k+1}$ , receive  $r_{k+1}$

$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$

$$[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

**until** terminal state

**end for**

# Exploration-exploitation tradeoff

- Essential condition for convergence to  $Q^*$ :  
all  $(x, u)$  pairs must be visited infinitely often
- ⇒ **Exploration** necessary:  
sometimes, choose actions randomly
- **Exploitation** of current knowledge is also necessary:  
sometimes, choose actions greedily:

$$u_k = \arg \max_{\bar{u}} Q(x_k, \bar{u})$$

# Exploration-exploitation tradeoff

- Essential condition for convergence to  $Q^*$ : all  $(x, u)$  pairs must be visited infinitely often
- ⇒ **Exploration** necessary:  
sometimes, choose actions randomly
- **Exploitation** of current knowledge is also necessary:  
sometimes, choose actions greedily:

$$u_k = \arg \max_{\bar{u}} Q(x_k, \bar{u})$$

Exploration-exploitation tradeoff crucial  
for performance of online RL

# Exploration-exploitation: $\varepsilon$ -greedy strategy

- Simple solution:  $\varepsilon$ -greedy

$$u_k = \begin{cases} \arg \max_{\bar{u}} Q(x_k, \bar{u}) & \text{with probability } (1 - \varepsilon_k) \\ \text{a random action} & \text{with probability } \varepsilon_k \end{cases}$$

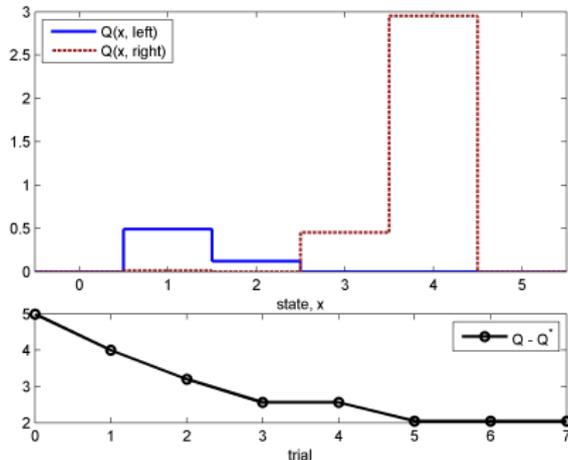
- Exploration probability  $\varepsilon_k \in (0, 1)$  is usually decreased over time

# Cleaning robot: Q-learning demo

Parameters:  $\alpha = 0.2$ ,  $\varepsilon = 0.3$  (constant)

$x_0 = 2$  or  $3$  (randomly)

Q-learning, trial 8, step 3



1 Learning paradigms

2 Elements of RL

3 Algorithms

Taxonomy

Q-learning

**SARSA**

4 Summary and outlook

# On-policy online RL: SARSA

On-policy: **find**  $Q^\pi$ , improve  $\pi$ , repeat

Similar to Q-learning:

- 1 Take Bellman equation for  $Q^\pi$ , at some  $(x, u)$ :

$$Q^\pi(x, u) = \rho(x, u) + \gamma Q^\pi(f(x, u), \pi(f(x, u)))$$

# On-policy online RL: SARSA

On-policy: **find**  $Q^\pi$ , improve  $\pi$ , repeat

Similar to Q-learning:

- 1 Take Bellman equation for  $Q^\pi$ , at some  $(x, u)$ :

$$Q^\pi(x, u) = \rho(x, u) + \gamma Q^\pi(f(x, u), \pi(f(x, u)))$$

- 2 Turn into iterative update:

$$Q(x, u) \leftarrow \rho(x, u) + \gamma Q(f(x, u), \pi(f(x, u)))$$

# On-policy online RL: SARSA

On-policy: **find**  $Q^\pi$ , improve  $\pi$ , repeat

Similar to Q-learning:

- 1 Take Bellman equation for  $Q^\pi$ , at some  $(x, u)$ :

$$Q^\pi(x, u) = \rho(x, u) + \gamma Q^\pi(f(x, u), \pi(f(x, u)))$$

- 2 Turn into iterative update:

$$Q(x, u) \leftarrow \rho(x, u) + \gamma Q(f(x, u), \pi(f(x, u)))$$

- 3 Use sample  $(x_k, u_k, r_{k+1}, x_{k+1}, u_{k+1})$  at each step  $k$ :

$$Q(x_k, u_k) \leftarrow r_{k+1} + \gamma Q(x_{k+1}, u_{k+1})$$

Note:  $u_{k+1} = \pi(f(x_k, u_k))$ ,  $\pi$  = **policy being followed**

## SARSA (cont'd)

- 4 Make update incremental:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot [r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

Note that

$$r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)$$

is the **temporal difference** here

$(x_k, u_k, r_{k+1}, x_{k+1}, u_{k+1}) =$

(**S**tate, **A**ction, **R**eward, **S**tate, **A**ction) = **SARSA**

# Complete SARSA algorithm

## SARSA

**for** every trial **do**

initialize  $x_0$ , choose initial action  $u_0$

**repeat** for each step  $k$

apply  $u_k$ , measure  $x_{k+1}$ , receive  $r_{k+1}$

choose **next** action  $u_{k+1}$

$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$

$$[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

**until** terminal state

**end for**

# Exploration-exploitation in SARSA

- For convergence—besides infinite exploration—SARSA requires **policy to eventually become greedy**
- E.g.,  $\varepsilon$ -greedy

$$u_k = \begin{cases} \arg \max_{\bar{u}} Q(x_k, \bar{u}) & \text{with probability } (1 - \varepsilon_k) \\ \text{a random action} & \text{with probability } \varepsilon_k \end{cases}$$

with  $\lim_{k \rightarrow \infty} \varepsilon_k = 0$

# Exploration-exploitation in SARSA

- For convergence—besides infinite exploration—SARSA requires **policy to eventually become greedy**
- E.g.,  $\varepsilon$ -greedy

$$u_k = \begin{cases} \arg \max_{\bar{u}} Q(x_k, \bar{u}) & \text{with probability } (1 - \varepsilon_k) \\ \text{a random action} & \text{with probability } \varepsilon_k \end{cases}$$

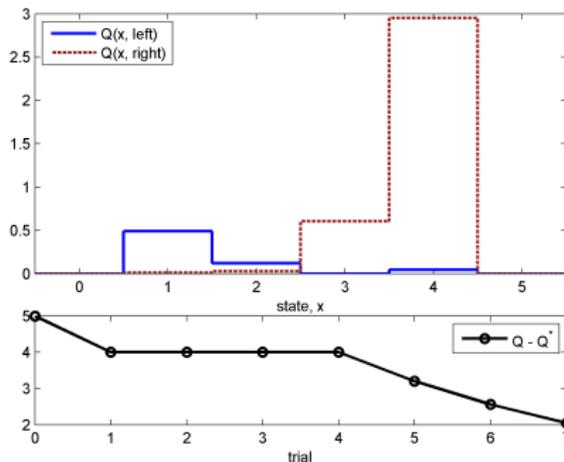
with  $\lim_{k \rightarrow \infty} \varepsilon_k = 0$

- Greedy actions  $\Rightarrow$  policy implicitly improved!  
(Recall **on-policy**: find  $Q^\pi$ , **improve**  $\pi$ , repeat)

# Cleaning robot: SARSA demo

Parameters like Q-learning:  $\alpha = 0.2$ ,  $\varepsilon = 0.3$  (constant)  
 $x_0 = 2$  or 3 (randomly)

SARSA, trial 8, step 3



# Summary

- **Reinforcement learning** = optimal, adaptive, model-free control
- Principle: reward signal as performance feedback
- Inspired from human and animal learning, but solid mathematical foundation
- Classical RL: small, discrete  $X$  and  $U$  (this lecture)

# A final look at the algorithms

Off-policy: Q-learning

On-policy: SARSA

# A final look at the algorithms

Off-policy: Q-learning

On-policy: SARSA

Typical parameter values:

- $\gamma$  0.9 or larger
- $\alpha_k$  under 0.5 or diminishing schedule
- $\epsilon_k$  around 0.1 or diminishing schedule

## Next lecture

Still to address:

- Continuous state and action spaces  $X, U$
- More algorithms: actor-critic, model-learning, etc.

Part II – RL using function approximation