Machine Learning for Systems and Control

5SC28

Lecture 5A

dr. ir. Maarten Schoukens & dr. ir. Roland Tóth

Control Systems Group

Department of Electrical Engineering

Eindhoven University of Technology

Academic Year: 2020-2021 (version 1.01)

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Learning Outcomes

Data-Driven Modelling → previous lectures

- **Explain, discuss, compare and interpret** the main techniques and theory for reinforcement learning based control starting from **classical Q-learning** up to **actor-critic** and **model internalization**-based methods;
- **Recommend and evaluate** a machine learning method for a real-life application in a systems and **control setting**.
- **Implement, tailor, and apply** the Gaussian process, (deep) neural network and **reinforcement learning techniques** for model and control learning on real-world systems (e.g. on an inverted pendulum laboratory setup).

Reinforcement Learning

What is Reinforcement Learning?

Multi-Armed Bandits

Finite Markov Decision Process

Reinforcement Learning

What is Reinforcement Learning?

Multi-Armed Bandits

Finite Markov Decision Process

What is Reinforcement Learning?

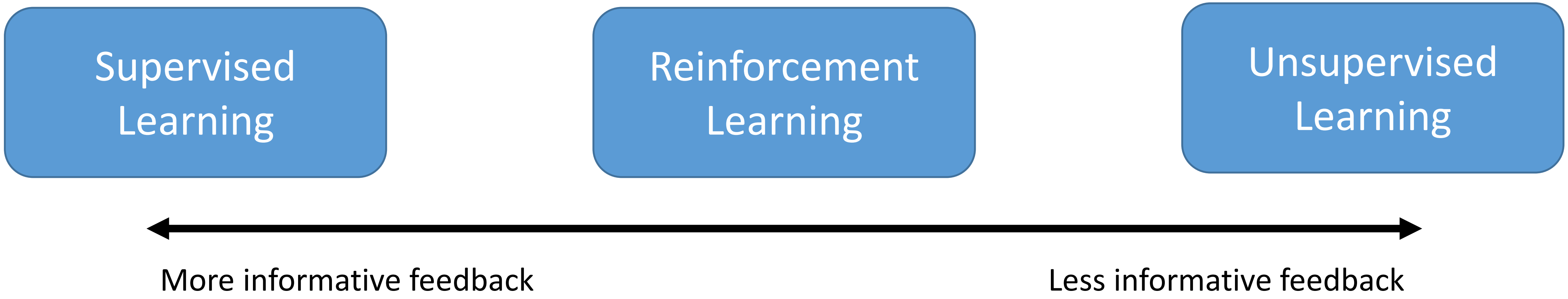
Learn by interacting with the environment

What is Reinforcement Learning?

Learn by interacting with the environment

Closed loop – awareness of the response to your action
Exploring: cause – effect
Goal directed learning
No explicit teacher

What is Reinforcement Learning?



What is Reinforcement Learning?

Supervised Learning

Reinforcement Learning

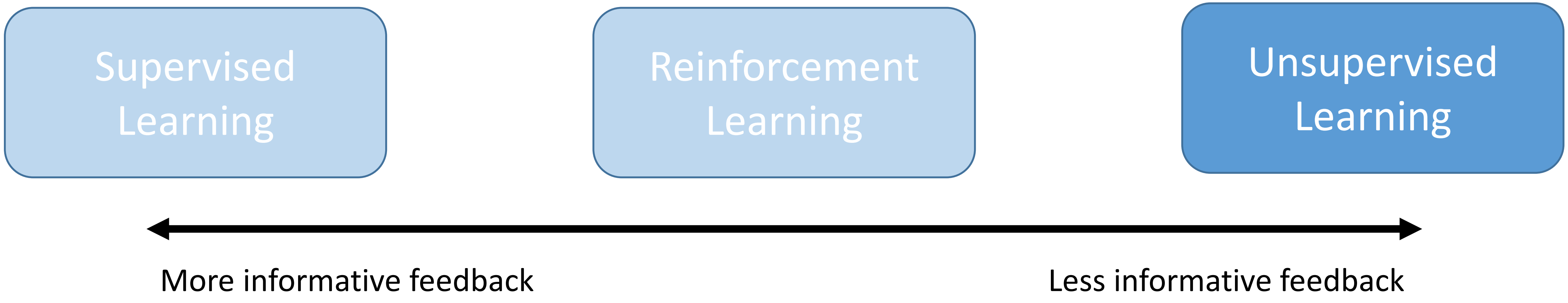
Unsupervised Learning

More informative feedback ⟷ Less informative feedback

Inputs and outputs are known
Infer input-output relationship

e.g. function estimation

What is Reinforcement Learning?



Only the inputs are known
Find patterns and features from data

e.g. clustering

What is Reinforcement Learning?

Supervised Learning

Reinforcement Learning

Unsupervised Learning

More informative feedback

Less informative feedback

Correct outputs not available, only rewards
Find optimal policy / behavior / controller

What is Reinforcement Learning?

"Reinforcement learning is a ***computational approach*** *to understanding and* ***automating goal-directed learning*** *and decision-making. It is distinguished from other computational approaches by its* ***emphasis on learning by an agent from direct interaction with its environment****, without relying on exemplary supervision or complete models of the environment."* [1]

[1] R.S. Sutton and A.G. Barto, "Reinforcement Learning: An Introduction", The MIT Press, 2018

What is Reinforcement Learning?

"Reinforcement learning is a ***computational approach*** *to understanding and* ***automating goal-directed learning*** *and decision-making. It is distinguished from other computational approaches by its* ***emphasis on learning by an agent from direct interaction with its environment****, without relying on exemplary supervision or complete models of the environment."* [1]

Reinforcement learning is about adaptive, model-free control

[1] R.S. Sutton and A.G. Barto, "Reinforcement Learning: An Introduction", The MIT Press, 2015

Reinforcement Learning

What is Reinforcement Learning?

Multi-Armed Bandits

Finite Markov Decision Process

One-Armed Bandit

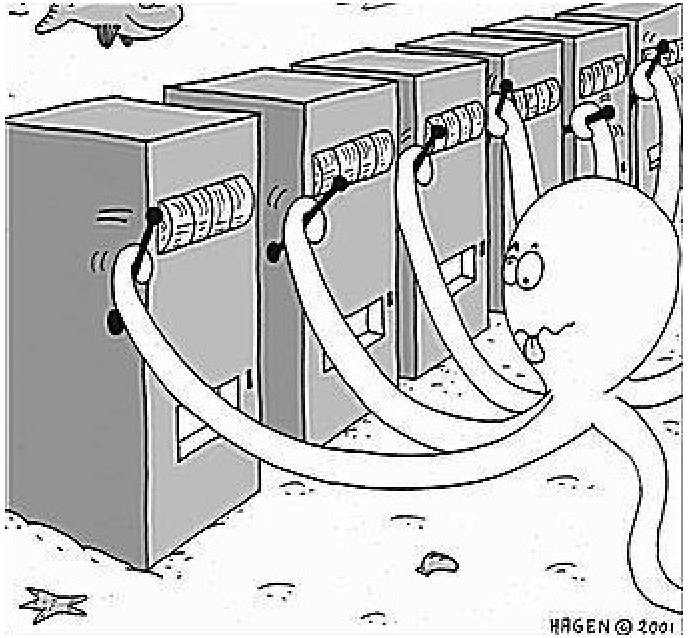
Source: https://www.partycity.com/

1. Pull the lever

2. Get a reward r with some probability $f_i(r)$

Multi-Armed Bandits

Multiple slot machines, each with their own unknown probability density function $f_i(r)$
You will play for a certain time (N lever pulls).

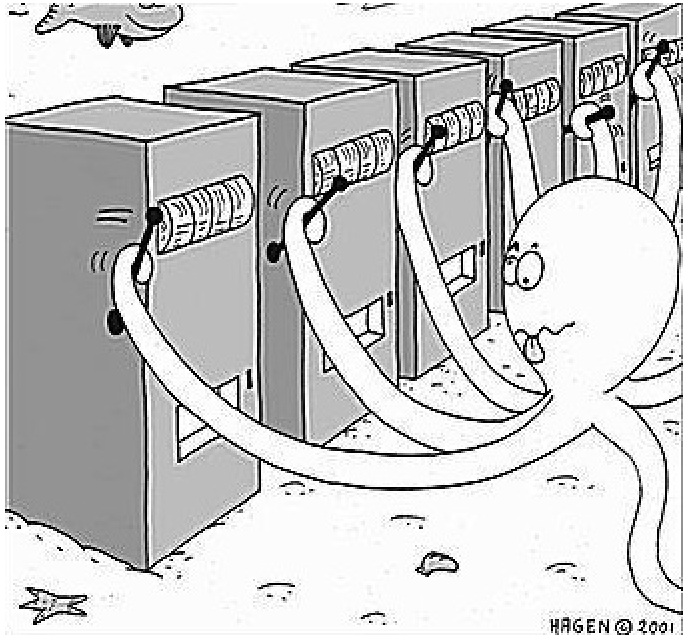


Source: https://www.analyticsvidhya.com/

Multi-Armed Bandits

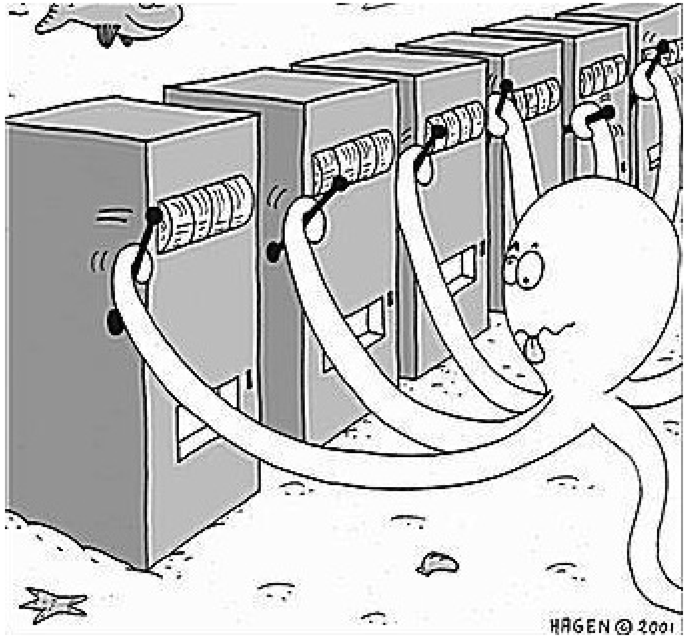
Multiple slot machines, each with their own unknown probability density function $f_i(r)$
You will play for a certain time (N lever pulls).

How to maximize the reward?
Which levers to pull?



Source: https://www.analyticsvidhya.com/

Multi-Armed Bandits



Source: https://www.analyticsvidhya.com/

Multiple slot machines, each with their own unknown probability density function $f_i(r)$
You will play for a certain time (N lever pulls).

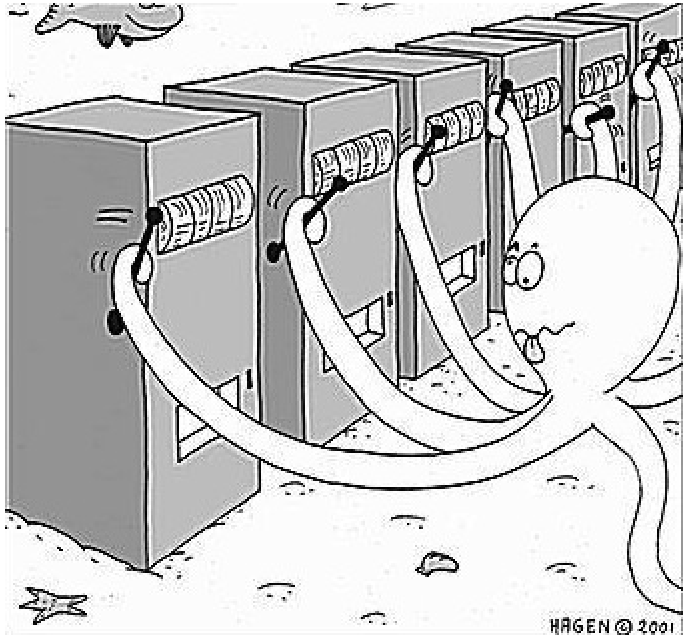
How to maximize the reward?
Which levers to pull?

Exploration vs Exploitation

Explore the reward given by the different slot machines?

Exploit the maximal reward currently known?

Practical Relevance



Source: https://www.analyticsvidhya.com/

Online Advertising

- Explore interests of user
- Exploit clicks of the user

Network Routing

- Allocate the best channel (maximize throughput)
- Monitor channel quality

Clinical Trials

- Try different treatments
- Give the best treatment

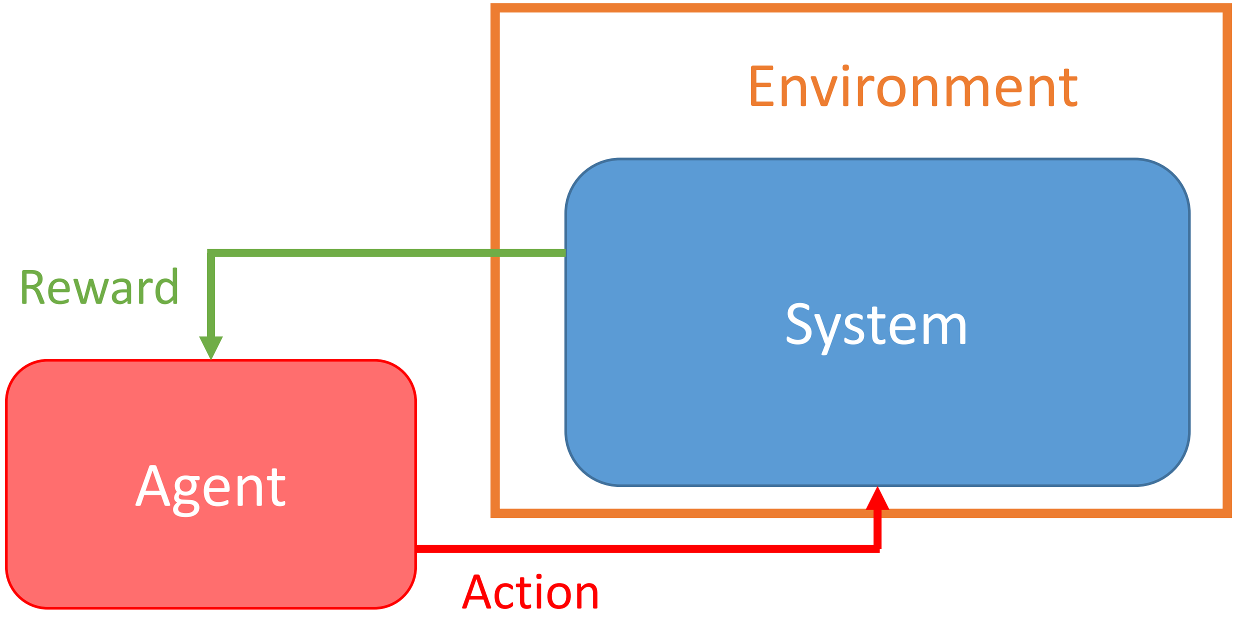
Going out for dinner

- Try a new place
- Go back to your favorite

Reinforcement Learning Relevance

Agent interacts with the environment through **Actions**

Receive **Reward** as a performance feedback



Action: Pull a lever
Reward: The return of the bandit

Goal: Maximize cumulative reward

Illustrates the exploration – exploitation trade-off

Problem Setting

K slot machines, each with their own unknown probability density function $f_i(r)$
You will play for a certain time (N lever pulls).

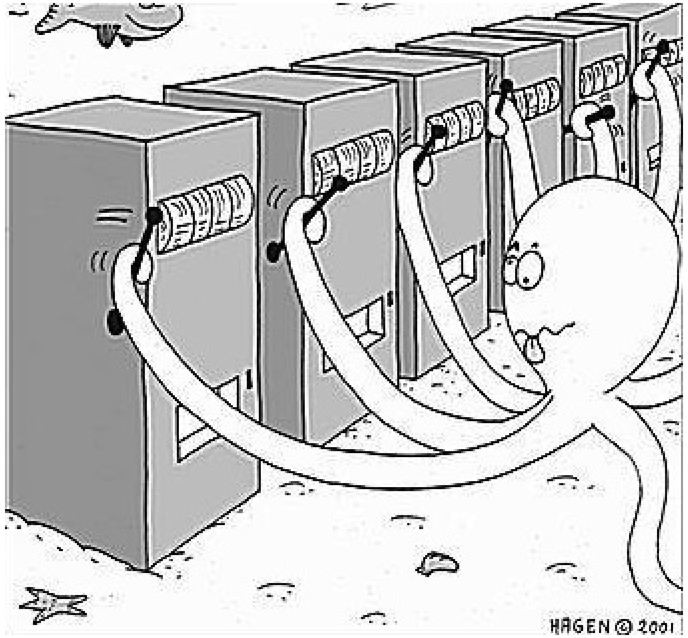
Rewards r: 0 or 1

P_i : probability of machine i resulting in a reward r = 1

The value of action $u \in U$ is the expected reward:

$$Q(u) = \mathbb{E}\{r|u\}$$

Goal: maximize cumulative reward $\sum_{k=1}^{N} r_k$



Source: https://www.analyticsvidhya.com/

Maximal Reward

Always pull the lever with the highest expected value

Optimal reward probability:

$$P^* = Q(u^*) = \max_{u \in U} Q(u) = \max_{1 \leq i \leq K} P_i$$

Goal

Always pull the lever with the highest expected value

Optimal reward probability:

$$P^* = Q(u^*) = \max_{u \in U} Q(u) = \max_{1 \le i \le K} P_i$$

Reformulated Goal: minimize cost function $$V_N = \mathbb{E}\left\{ \sum_{k=1}^{N} \underbrace{(P^* - Q(u_k))}_{\text{regret}} \right\}$$

Problem: P_i is unknown!

Exploitation - Exploration

Goal: minimize cost function $V_N = \mathbb{E}\left\{\sum_{k=1}^{N} \underbrace{(P^* - Q(a_k))}_{\text{regret}}\right\}$

Problem: P_i is unknown!

Dilemma:

1. Minimize immediate regret at time t (greedy action / exploitation)
2. Improve estimates of P_i (exploration)

Action Value Function

Define the action value function at time t as:

$$Q_k(u) = \frac{\sum_{i=1}^{k} r_k 1_{u_k=u}}{\sum_{i=1}^{k} 1_{u_k=u}} \Rightarrow \frac{\text{Total reward obtained with action } u}{\text{Total times action } u \text{ is chosen}}$$

This function is an estimate of $Q(u)$ using the observations available at time k.

Strategy 1: Greedy Approach

$Q_0(u)$ is set at a user-chosen initial value.

Always choose the action for which $Q_k(u)$ is maximal: $u_{k+1} = \arg\max_u Q_k(u)$

Strategy 1: Greedy Approach

$Q_0(u)$ is set at a user-chosen initial value.

Always choose the action for which $Q_k(u)$ is maximal: $u_{k+1} = \arg\max_u Q_k(u)$

No exploration!

Strategy 2: ε - Greedy Approach

$Q_0(u)$ is set at a user-chosen initial value.

Choose the action for which $Q_k(u)$ is maximal with probability $(1 - \epsilon)$

Sample randomly one of the other actions with probability ϵ

Strategy 2: ε - Greedy Approach

$Q_0(u)$ is set at a user-chosen initial value.

Choose the action for which $Q_k(u)$ is maximal with probability $(1-\epsilon)$

Sample randomly one of the other actions with probability ϵ

Exploration!

Strategy 2: ε - Greedy Approach

$Q_0(u)$ is set at a user-chosen initial value.

Choose the action for which $Q_k(u)$ is maximal with probability $(1-\epsilon)$

Sample randomly one of the other actions with probability ϵ

Exploration!

But we keep exploring even long after the estimates $Q_k(u)$ converged to their expected values $Q(u)$

Recursive Implementation

If action u is taken:

Action result

$$Q_k(u) = \frac{\sum_{i=1}^{k} r_k 1_{u_k=u}}{\sum_{i=1}^{k} 1_{u_k=u}} = Q_{k-1}(u) + \frac{1}{N_u}(r_k - Q_{k-1}(u))$$

Number of times
action u is taken

Recursive Implementation

If action u is taken:

$$Q_k(u) = \frac{\sum_{i=1}^{k} r_k 1_{u_k=u}}{\sum_{i=1}^{k} 1_{u_k=u}} = Q_{k-1}(u) + \frac{1}{N_u}(r_k - Q_{k-1}(u))$$

$$X_k = \frac{1}{k}\sum_{i=1}^{k} x_i$$

$$=$$

Exercise 1a

$$=$$

$$= X_{k-1} + \frac{1}{k}\left(x_k - X_{k-1}\right)$$

Strategy 2: ε - Greedy Algorithm

Parameters: exploration probability $\epsilon \in [0, 1]$
Initialize $Q_0(u) = 0,\ N_u = 0$

Repeat forever

$$u \leftarrow \begin{cases} \arg\max_u Q(u) & \text{with probability } 1 - \epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

Exploitation

Exploration

Take action u and observe the reward r

$$N_u \leftarrow N_u + 1$$

$$Q(u) \leftarrow Q(u) + \frac{1}{N_u}(r - Q(u))$$

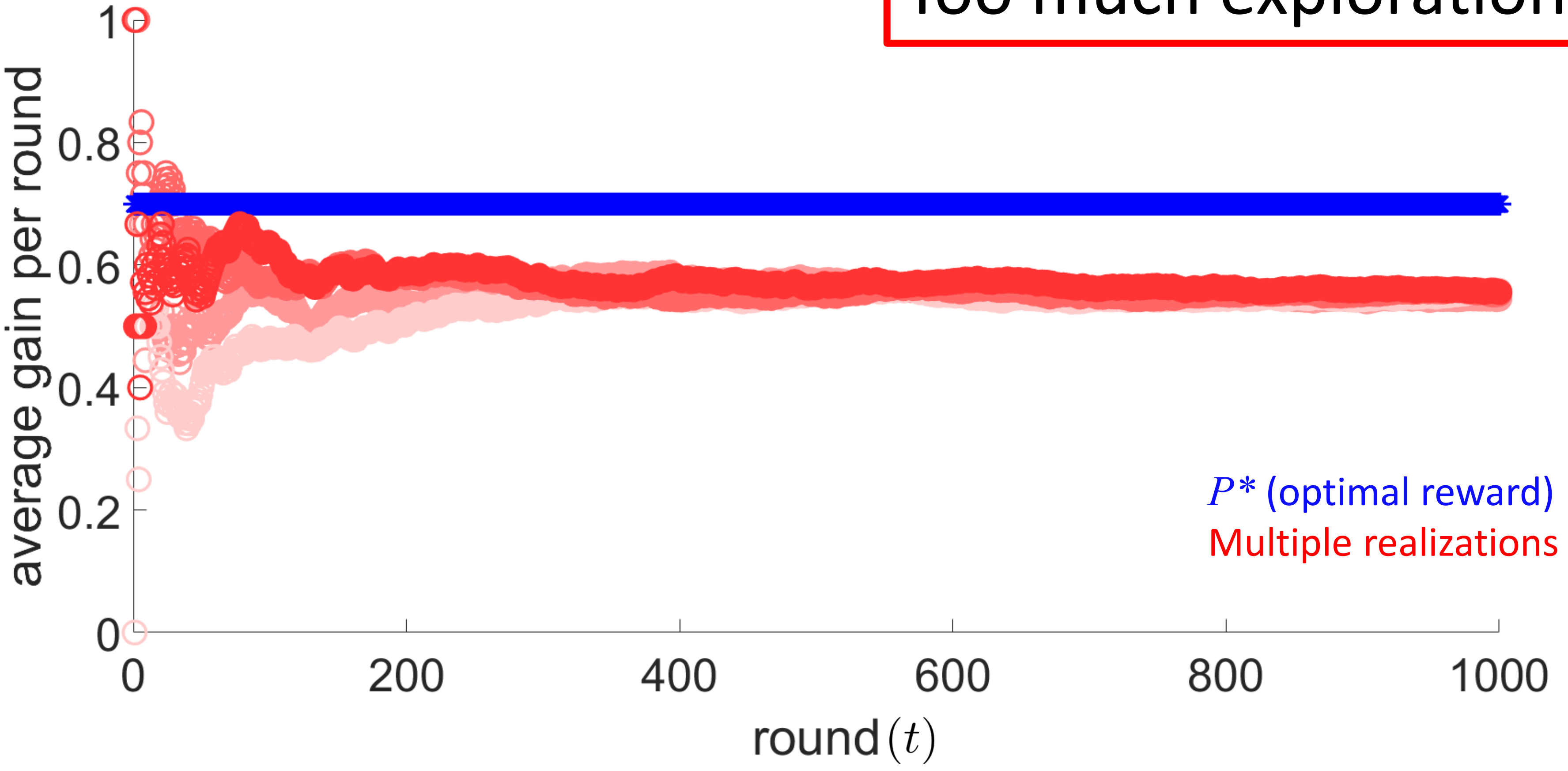
Example

$$P_1 = 0.5 \qquad P_2 = 0.4 \qquad P_3 = 0.7 \qquad P_4 = 0.6$$

Example: Random Choices

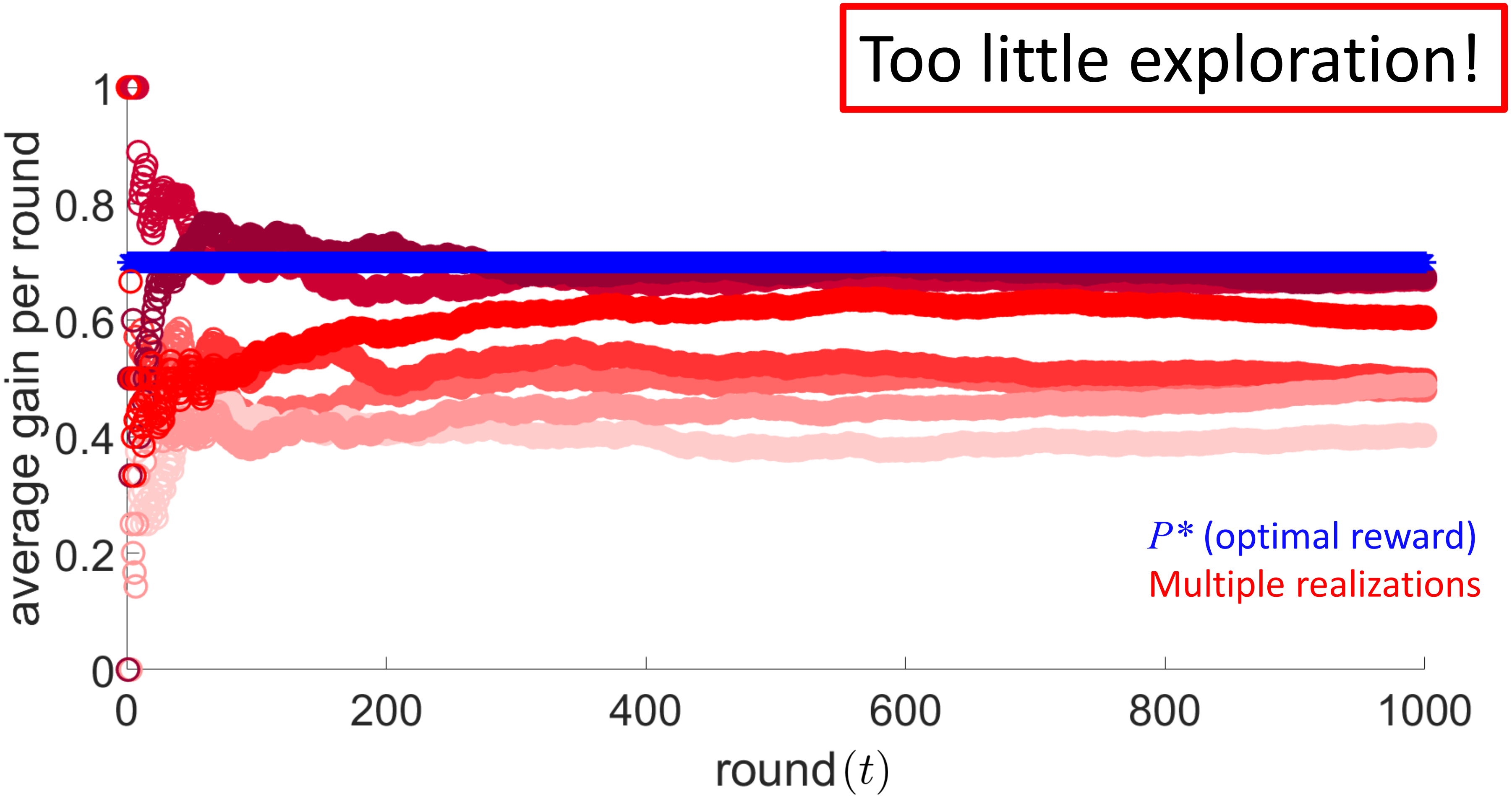
Too much exploration!

$$\frac{1}{k}\sum_{j=1}^{k} r_k$$



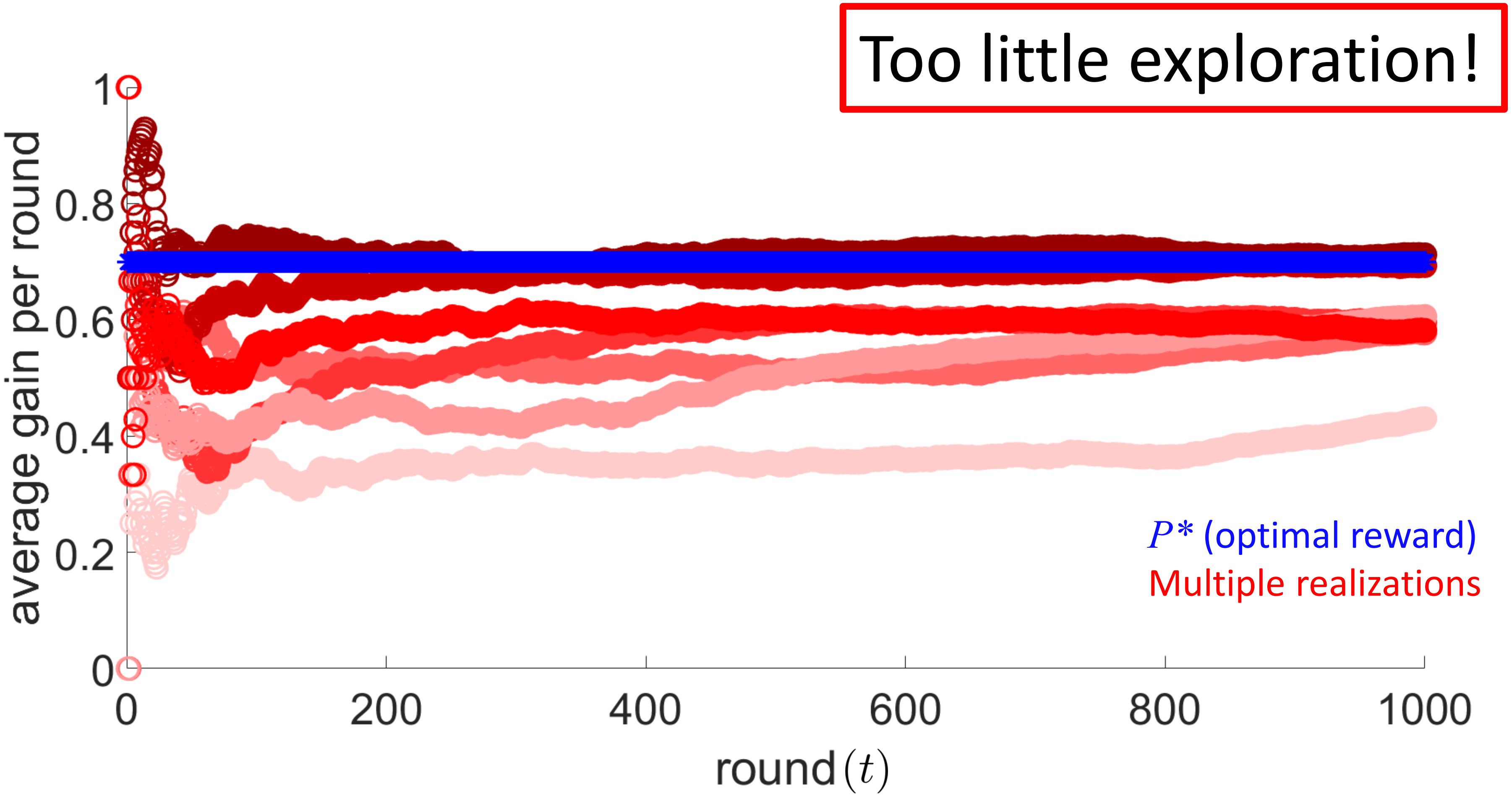
Example: Greedy Choice

$$\frac{1}{k}\sum_{j=1}^{k} r_k$$



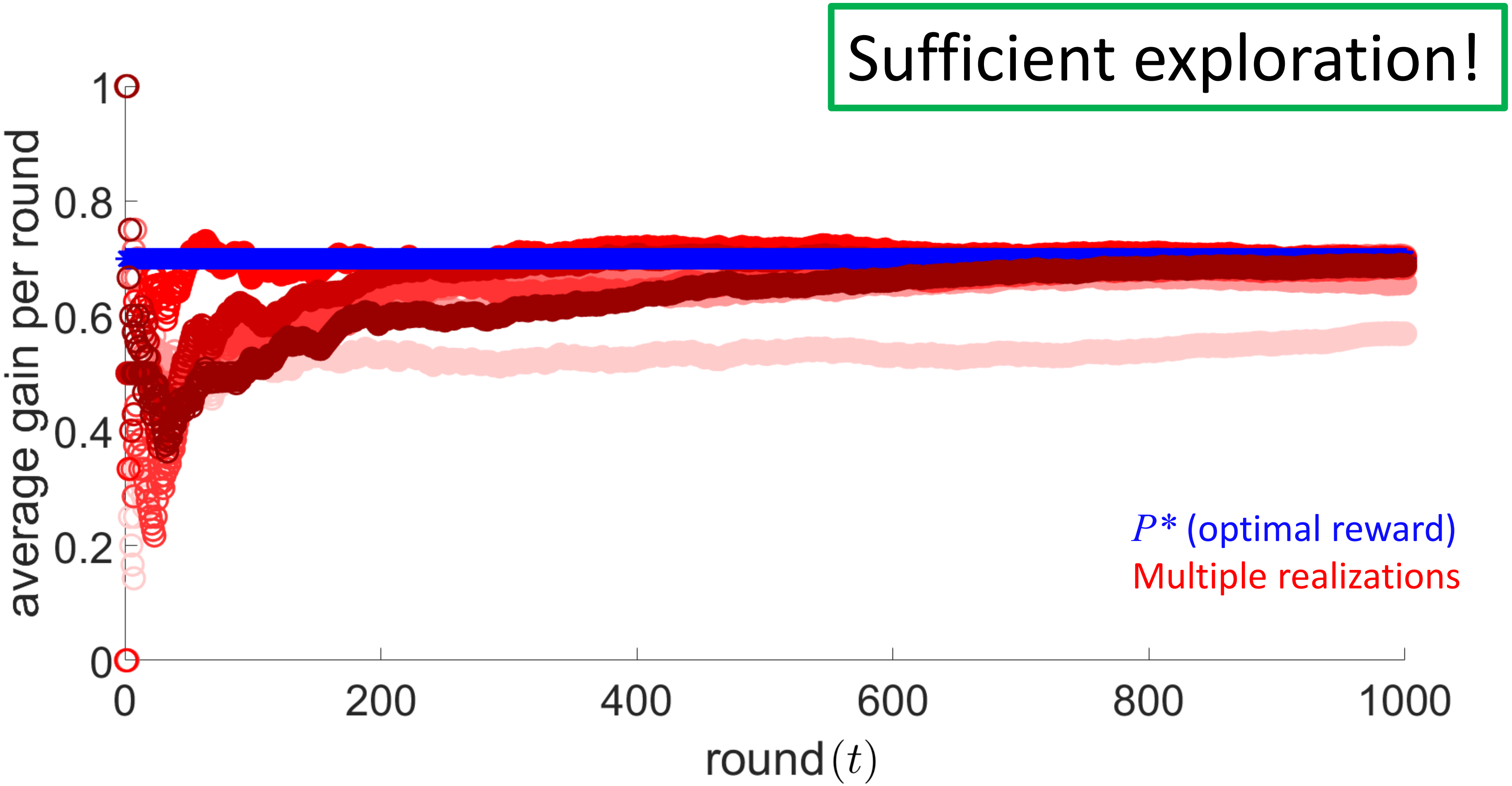
Example: ε - Greedy Choice (ε = 0.01)

$$\frac{1}{k}\sum_{j=1}^{k} r_k$$



Example: ε - Greedy Choice (ε = 0.1)

$$\frac{1}{t}\sum_{j=1}^{t} r_t$$



Reinforcement Learning

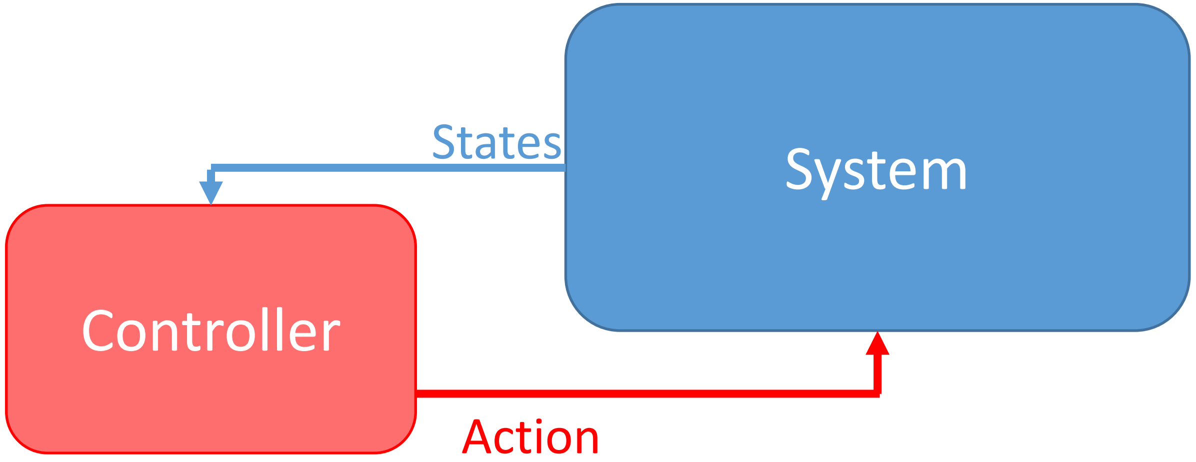
What is Reinforcement Learning?

Multi-Armed Bandits

The Reinforcement Learning Problem

Key Ingredients

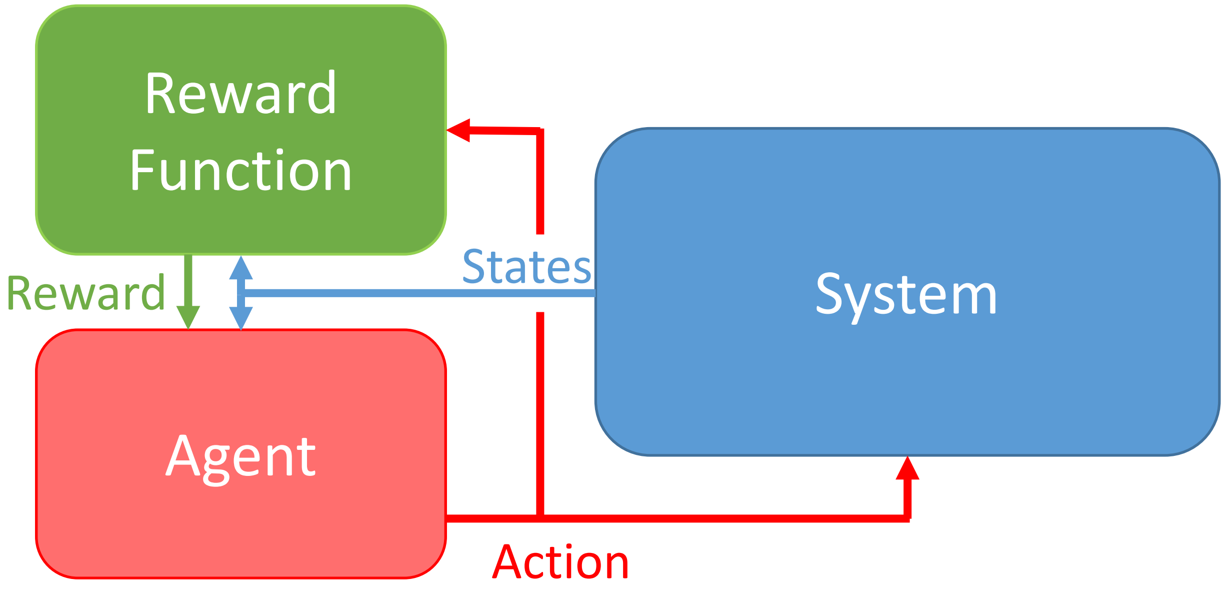
Controller interacts with the system through **States** and **Actions**



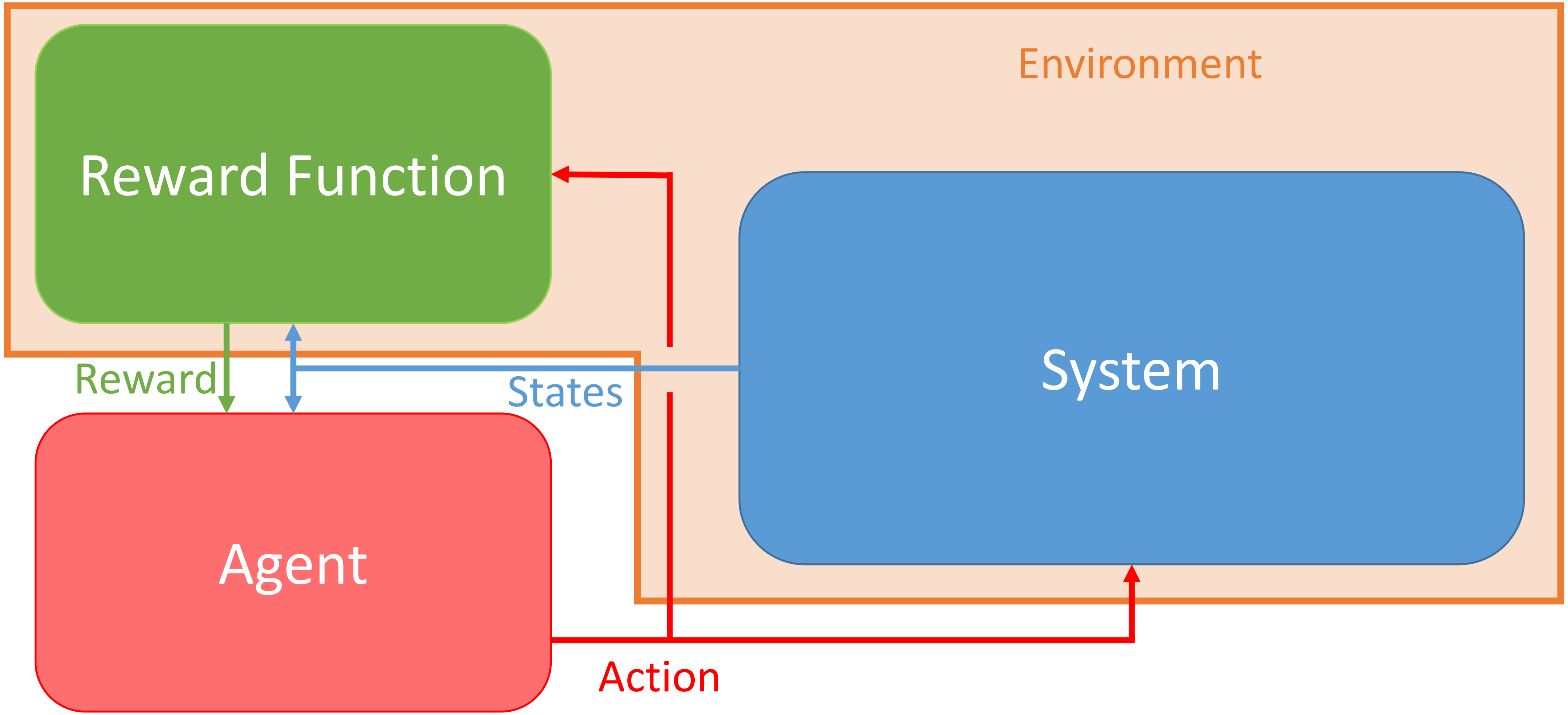
Key Ingredients

Agent interacts with the system through **States** and **Actions**

Receive **Reward** as a performance feedback

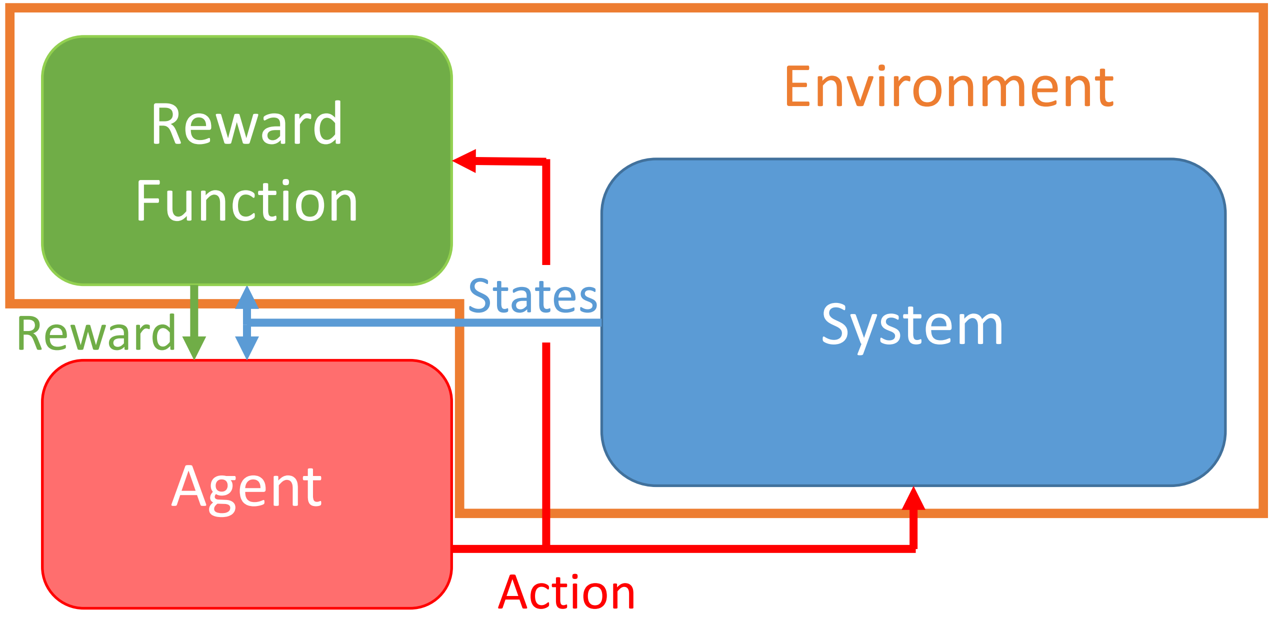


Key Ingredients



States: x Action: u Reward: r

Agent



The **Agent** is a state feedback controller:

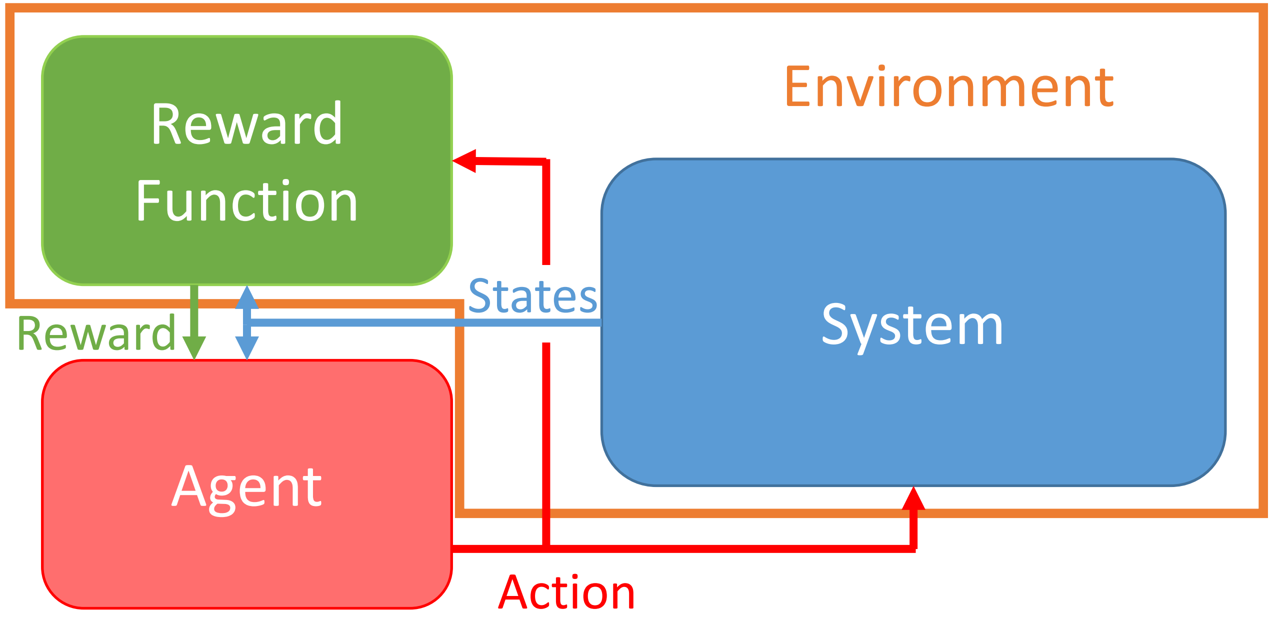
Learns optimal mapping from **States** to **Action**

Policy $\pi : X \rightarrow U$ is the control law

X is the finite state space

U is the finite action space

Goals and Rewards



Goal can be formalized as the maximization of the expected value of the cumulative sum of a received scalar signal (**Reward** signal)

Maximize the Expected Reward

Goals and Rewards

Finite Time Horizon: Expected Cumulative Return

$$G_k = \mathbb{E}_\pi \left\{ \sum_{j=k+1}^{N} r_j \right\}$$

Becomes infinity for t growing to infinity

Infinite Time Horizon: Expected Discounted Return

$$\boxed{G_k = \mathbb{E}_\pi \left\{ \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \right\}}$$

$$0 \leq \gamma < 1$$

Goals and Rewards

Infinite Time Horizon: Expected Discounted Return

$$G_k = \mathbb{E}_\pi \left\{ \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \right\} \qquad 0 \leq \gamma < 1$$

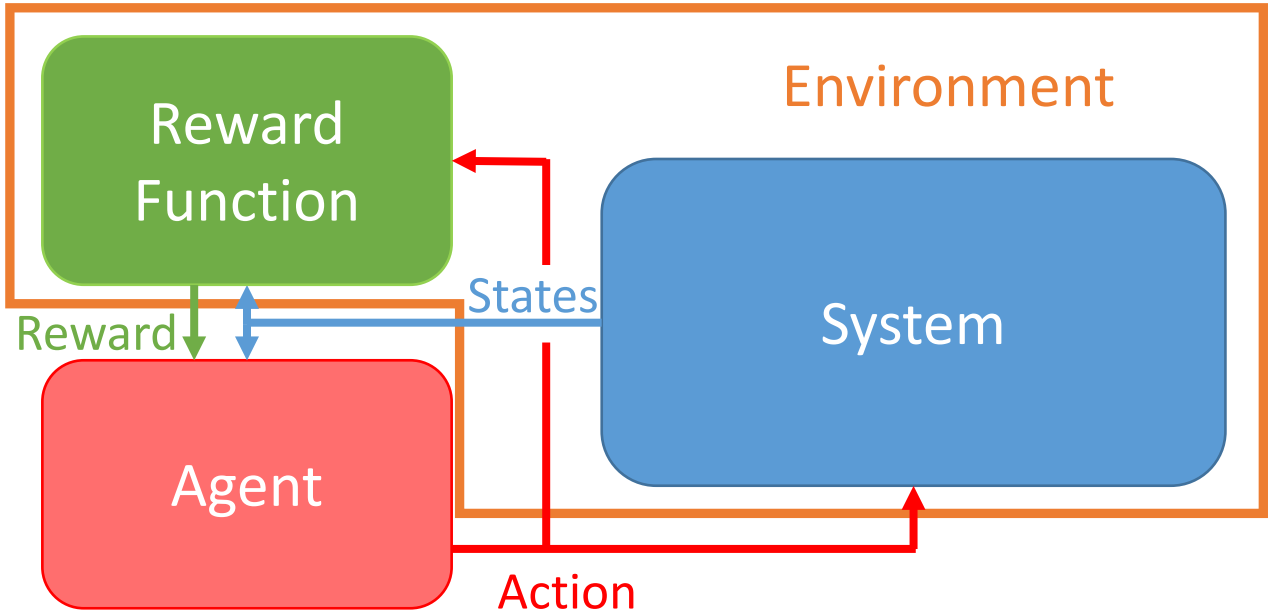
$\gamma = 0$ only care about immediate reward,

$\gamma \approx 1$ future rewards are strongly accounted for

Bounds infinite sum

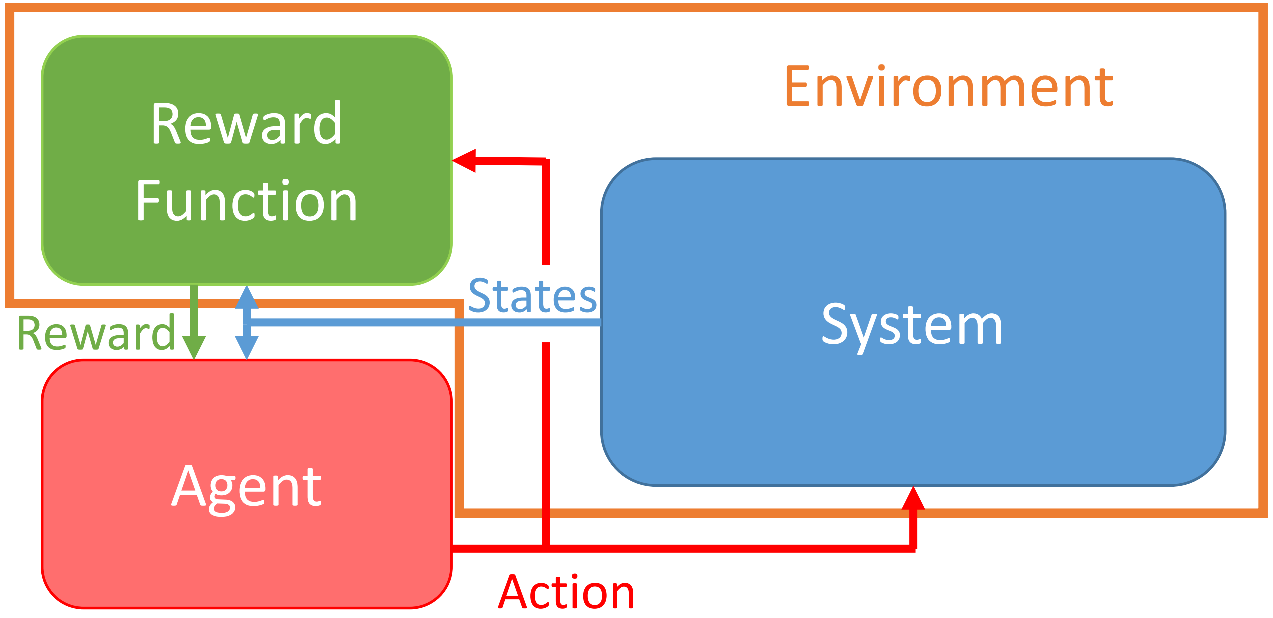
Encodes increasing uncertainty on the future

Environment



The **Agent** interacts with the **Environment** at a sequence of discrete time steps. Based on its **Actions** the **Agent** receives some representation of the **System State** and a numerical **Reward**.

Environment



The **Environment** is represented by a Markov Decision Process (MDP)

Environment (deterministic)

A finite **Markov Decision Process (MDP)** is a tuple $\langle X,\ U,\ g,\ \rho \rangle$ where:

X is the finite state space

U is the finite action space

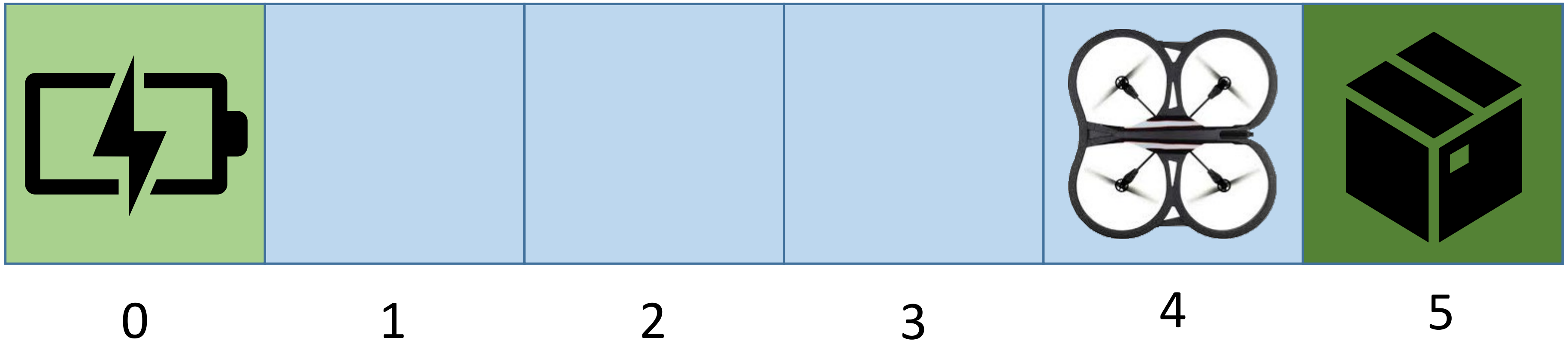
$g \ : \ X \times U \to X$ is the state transition function

$\rho \ : \ X \times U \to \mathbb{R}$ is the reward function

$x_{k+1} = g(x_k, u_k),\ r_{k+1} = \rho(x_k, u_k)$ with k being the discrete time

A **finite MDP** has finite state, action and reward sets.

Example (deterministic)

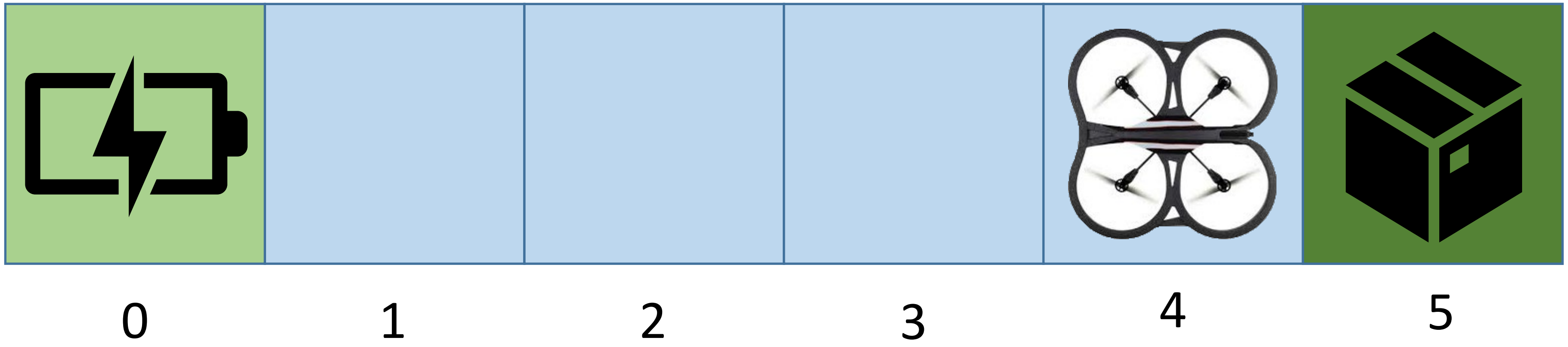


1D Package Delivery Drone

Goal: pick up package (+5) or power up (+1)

Episode terminates after reaching one of the goals

Example (deterministic)



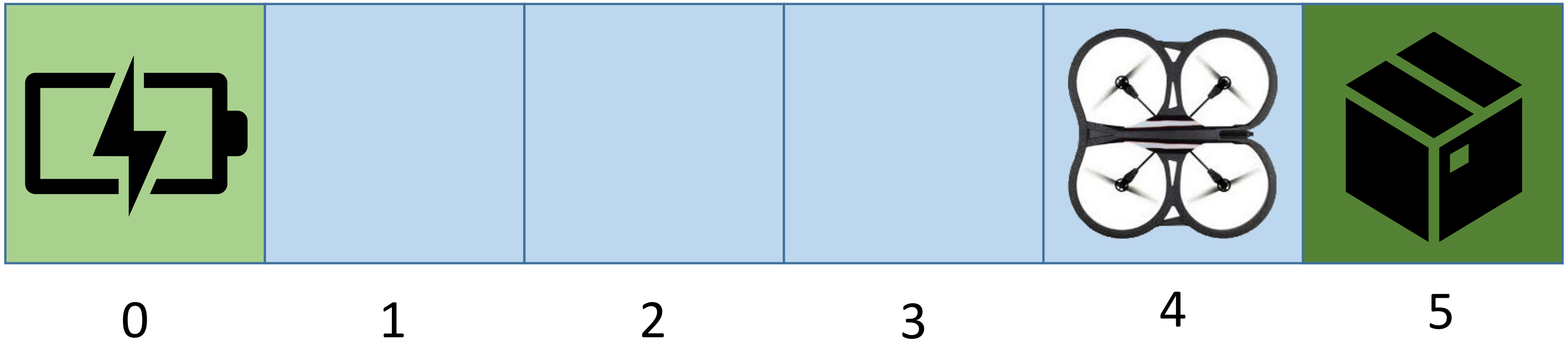
State: Drone Position

Actions: Move Left, Move Right

Rewards: Collect Package (+5), Power Up (+1)

$$X = \{0, 1, 2, 3, 4, 5\}$$
$$U = \{-1, 1\}$$

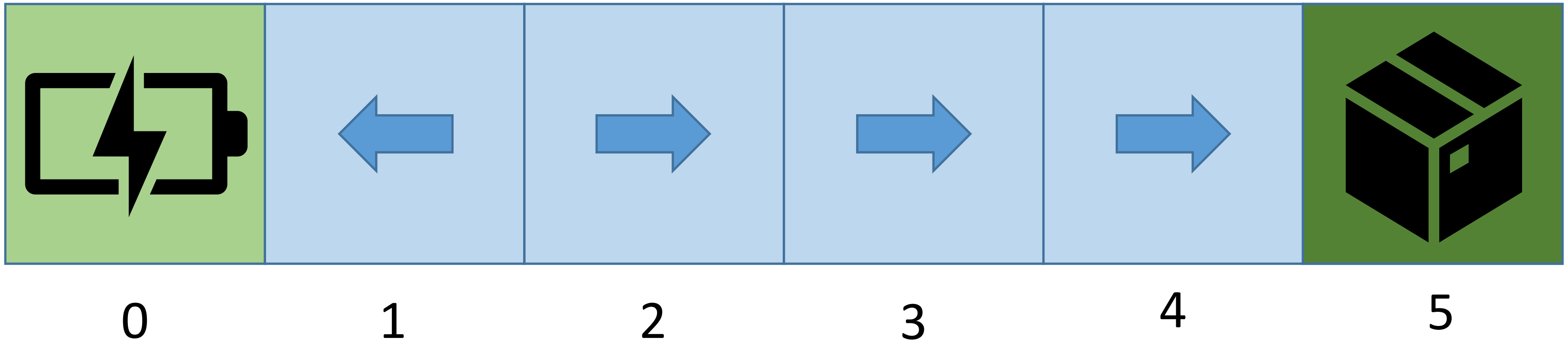
Example (deterministic)



State Transition Function $x_{k+1} = g(x_k, u_k) = \begin{cases} x_k & \text{if } x \text{ is terminal (0 or 5)} \\ x_k + u_k & \text{otherwise} \end{cases}$

Reward Function $r_k = \rho(x_k, u_k) = \begin{cases} 1 & \text{if } x = 1 \,\&\, u = -1 \quad \text{(power up)} \\ 5 & \text{if } x = 4 \,\&\, u = 1 \quad \text{(package)} \\ 0 & \end{cases}$

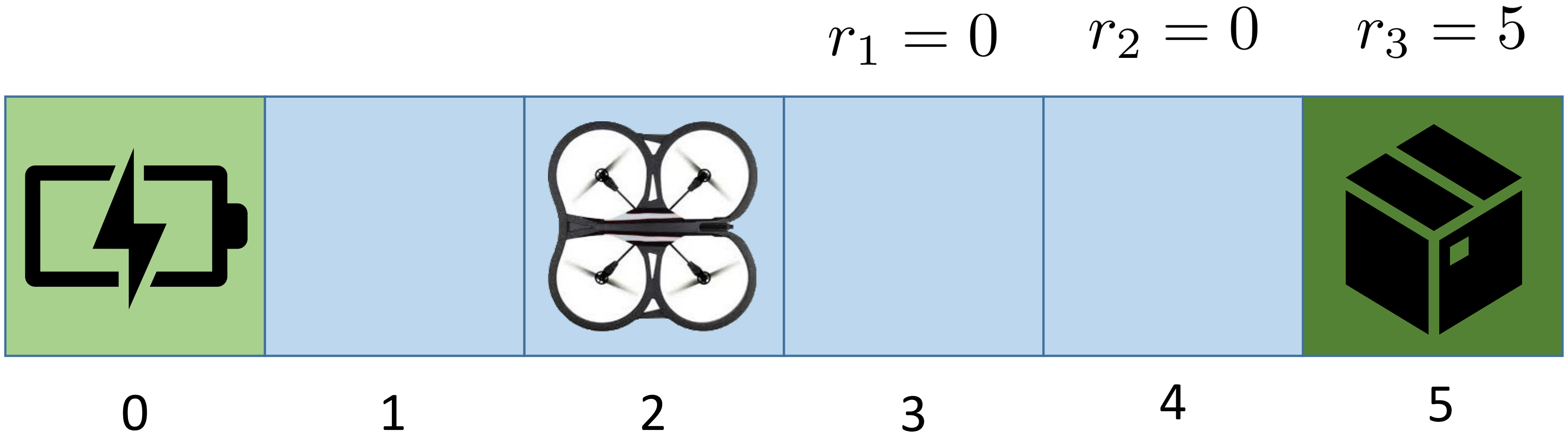
Example (deterministic)



Policy

$$\pi(0) = * \qquad \pi(2) = 1 \qquad \pi(4) = 1$$
$$\pi(1) = -1 \qquad \pi(3) = 1 \qquad \pi(5) = *$$

Example (deterministic)



Policy

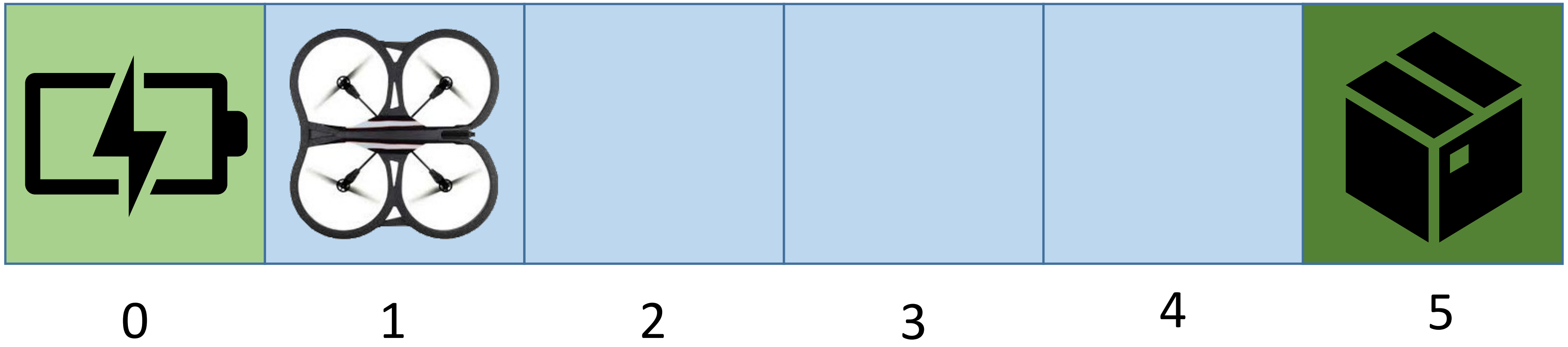
$$\pi(0) = * \quad \pi(2) = 1 \quad \pi(4) = 1$$
$$\pi(1) = -1 \quad \pi(3) = 1 \quad \pi(5) = *$$

Discounted reward

$$\gamma = 0.5 \qquad G_k = \mathbb{E}_\pi \left\{ \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \right\} = 1 \times 0 + 0.5 \times 0 + 0.5^2 \times 5$$

Example (deterministic)

$r_1 = 1$



Policy

$\pi(0) = *$ $\quad \pi(2) = 1$ $\quad \pi(4) = 1$

$\pi(1) = -1$ $\quad \pi(3) = 1$ $\quad \pi(5) = *$

Discounted reward

$\gamma = 0.5$

$$G_k = \mathbb{E}_\pi \left\{ \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \right\} = 1 \times 1$$

Reinforcement Learning – After the Break

Stochastic Environment

Towards Reinforcement Learning:

Value Function, Q-Function, Bellman Equation

Temporal Difference Learning

Q-Learning