VAIA FLAMES course: Reinforcement learning

16 January 2025

prof. dr. Pieter Libin
Artificial intelligence lab
Department of computer science
Vrije Universiteit Brussel

Introduction

Pieter.Libin@vub.be
@PieterLibin

ai.vub.ac.be
@aibrussels

- PhD in Computer science, AI lab, VUB
- Postdoc at Data science institute, UHasselt
- Assistant Professor, AI lab, VUB

- Research: decision making using realistic models and RL
- Interests: computational biology, big data, engineering

- Material: https://github.com/plibin/vala-rl-25

Course objectives

- **Intuition** about reinforcement learning
- How can *you* use reinforcement learning
- **Understanding** important reinforcement learning **algorithms**
- Get **hands-on experience** with deep RL algorithms in python

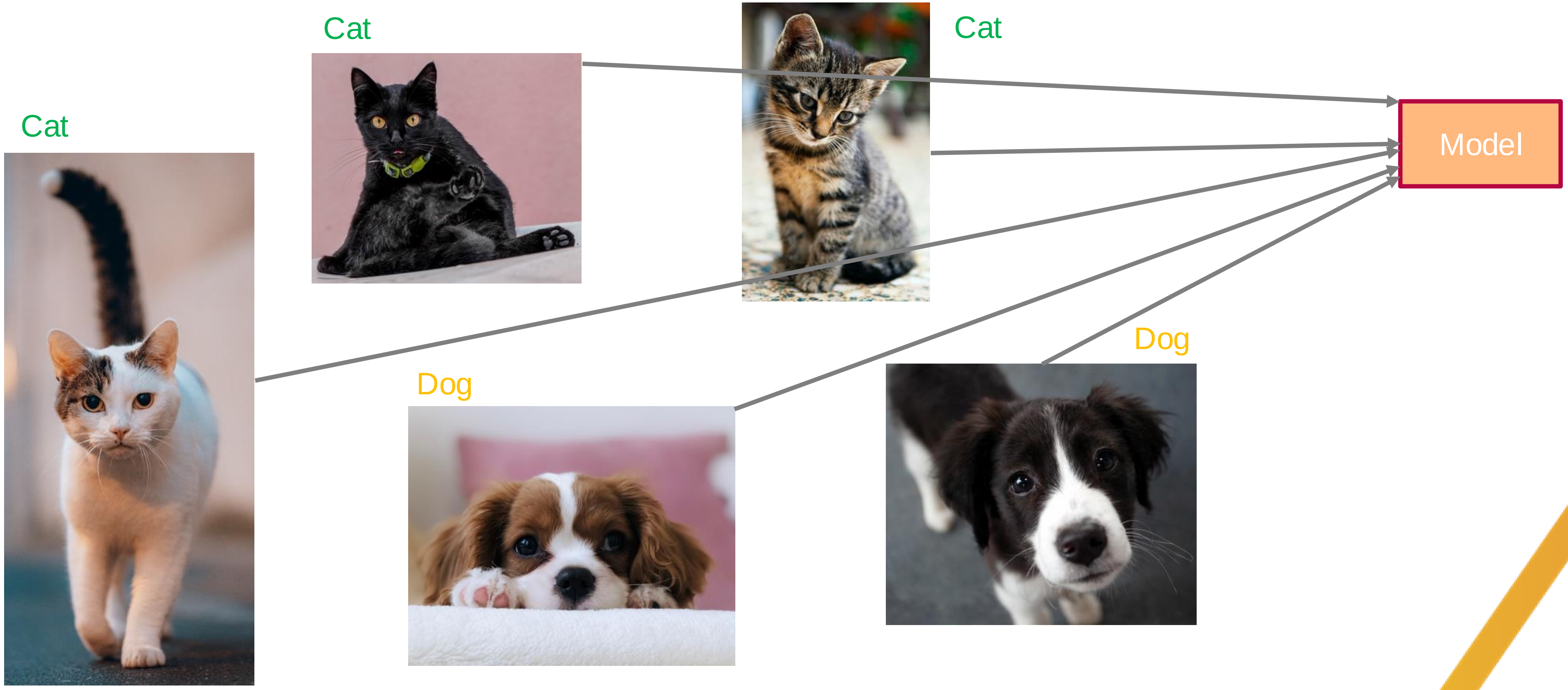
Question

- Did you already hear about RL and the application of it on the GO board game?

Reinforcement learning, a bit different

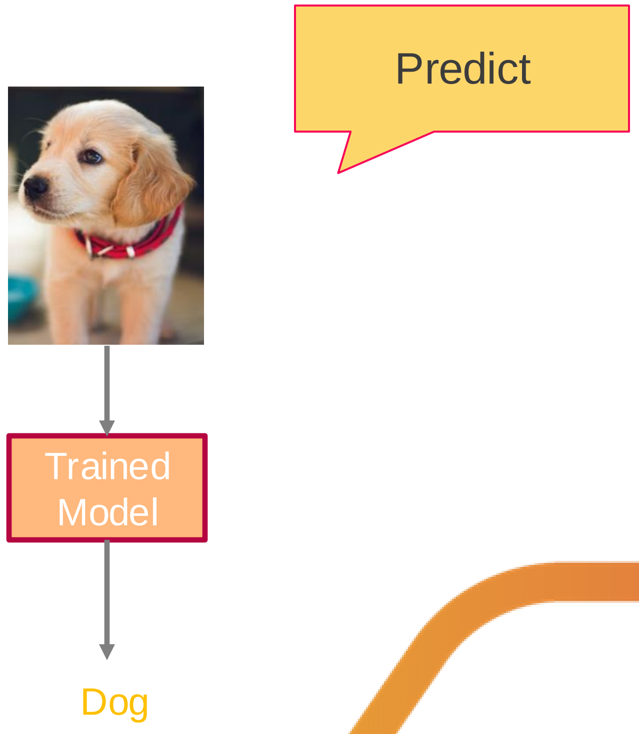
Train

- **Supervised** vs unsupervised machine learning
    - Learn from labelled vs unlabelled data



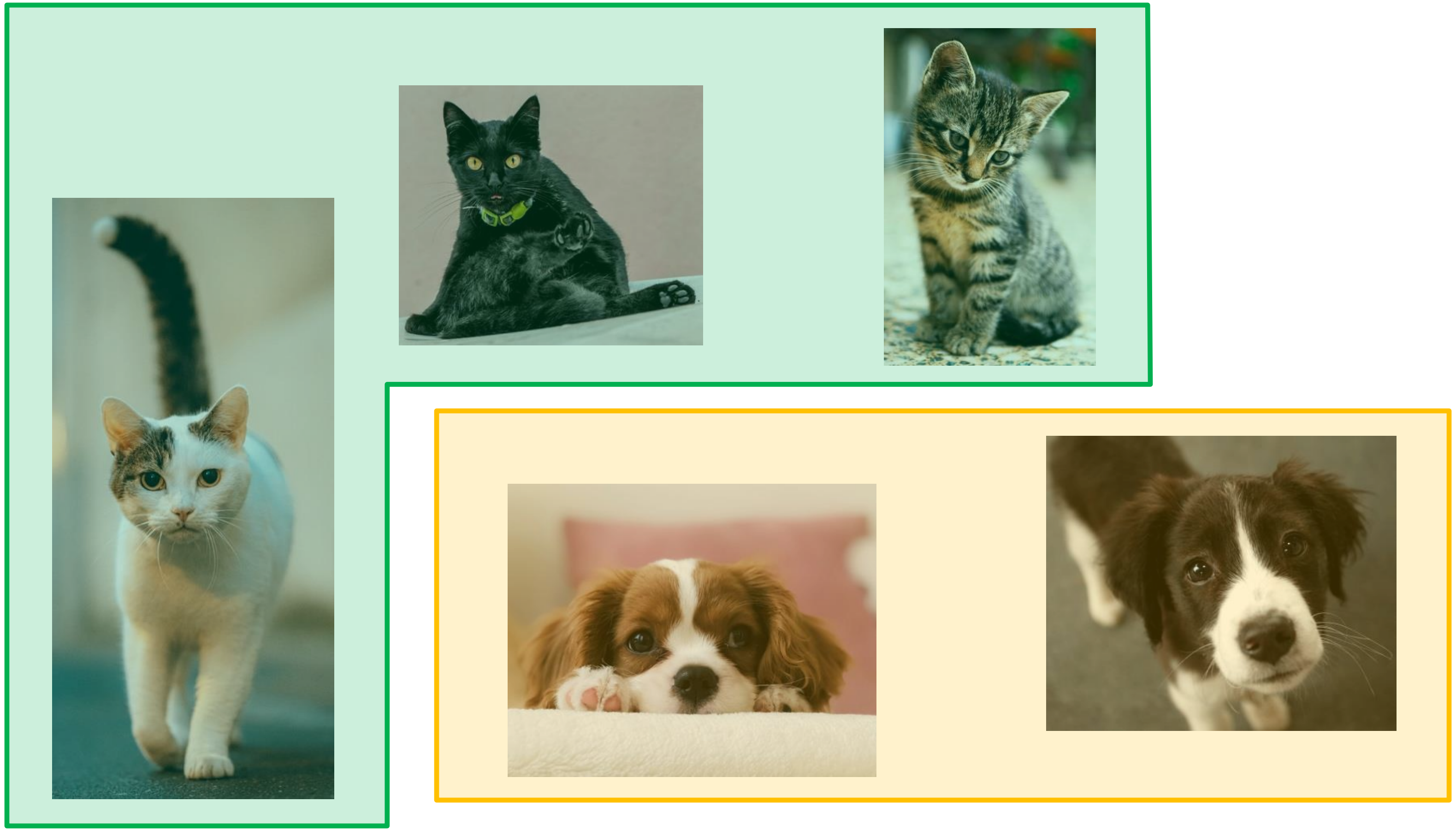
Reinforcement learning, a bit different

- **Supervised** vs unsupervised machine learning
  - Learn from labelled vs unlabelled data



Reinforcement learning, a bit different

- Supervised vs **unsupervised** machine learning
  - Learn from labelled vs unlabelled data

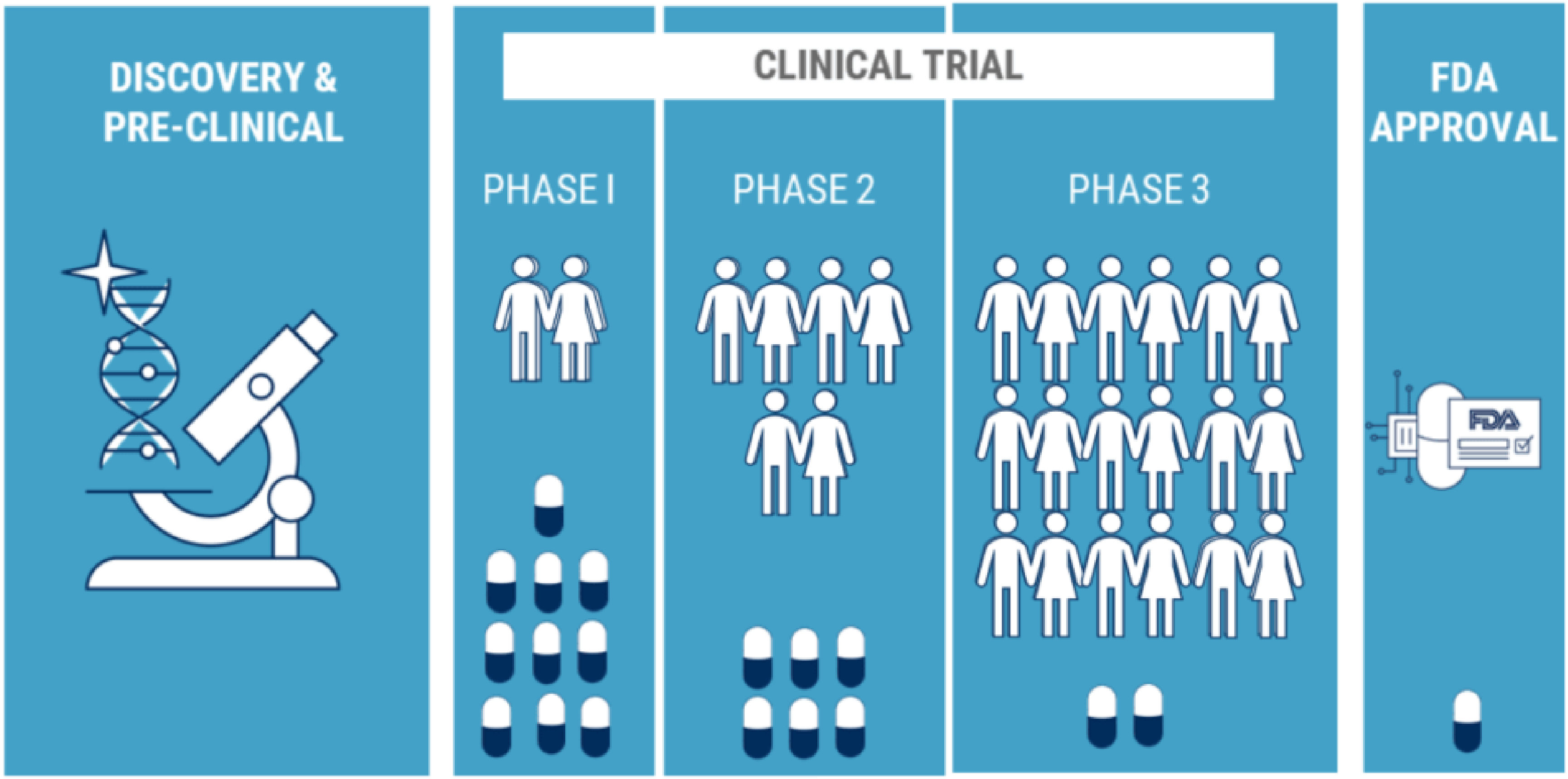


Reinforcement learning, a bit different

- Reinforcement learning:
  - sequential decision making problems
  - optimize behaviour by interacting with an environment
  - get to know an environment by trial-and-error
  - maximise long-term reward
- We have an **agent** that wants to learn an optimal **policy**, by inter**acting** with an **environment,** by maximising a **reward** signal

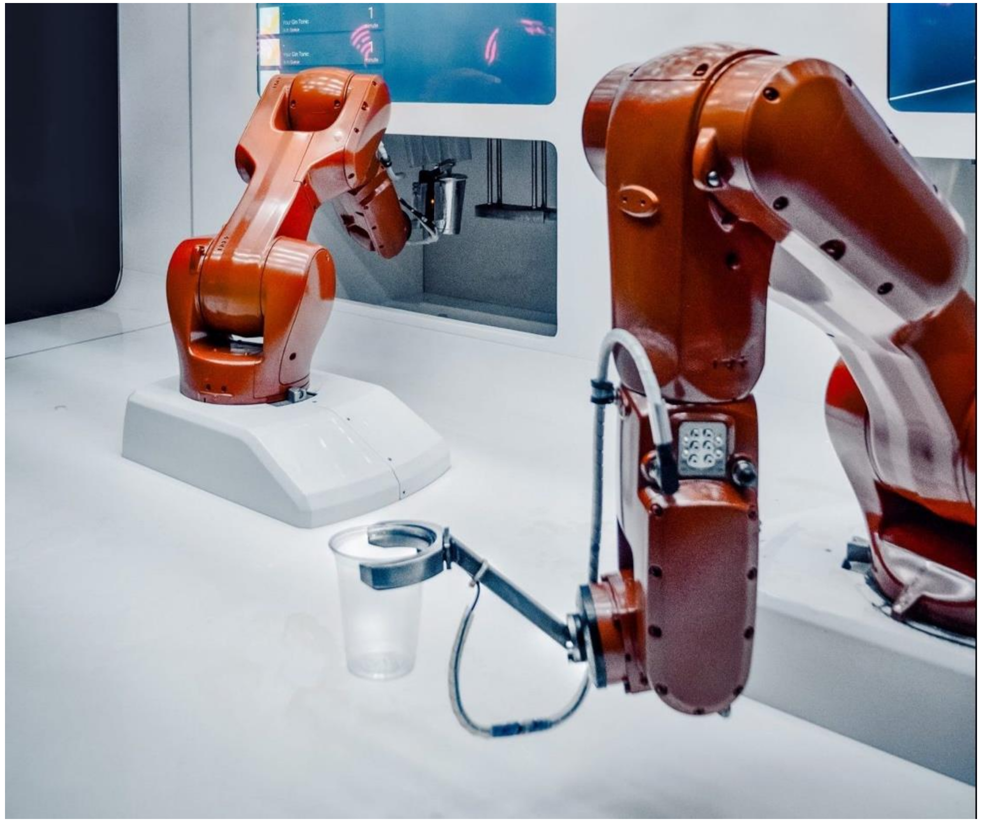
Reinforcement learning, some applications

- Design of clinical trials, Williamson (2019)



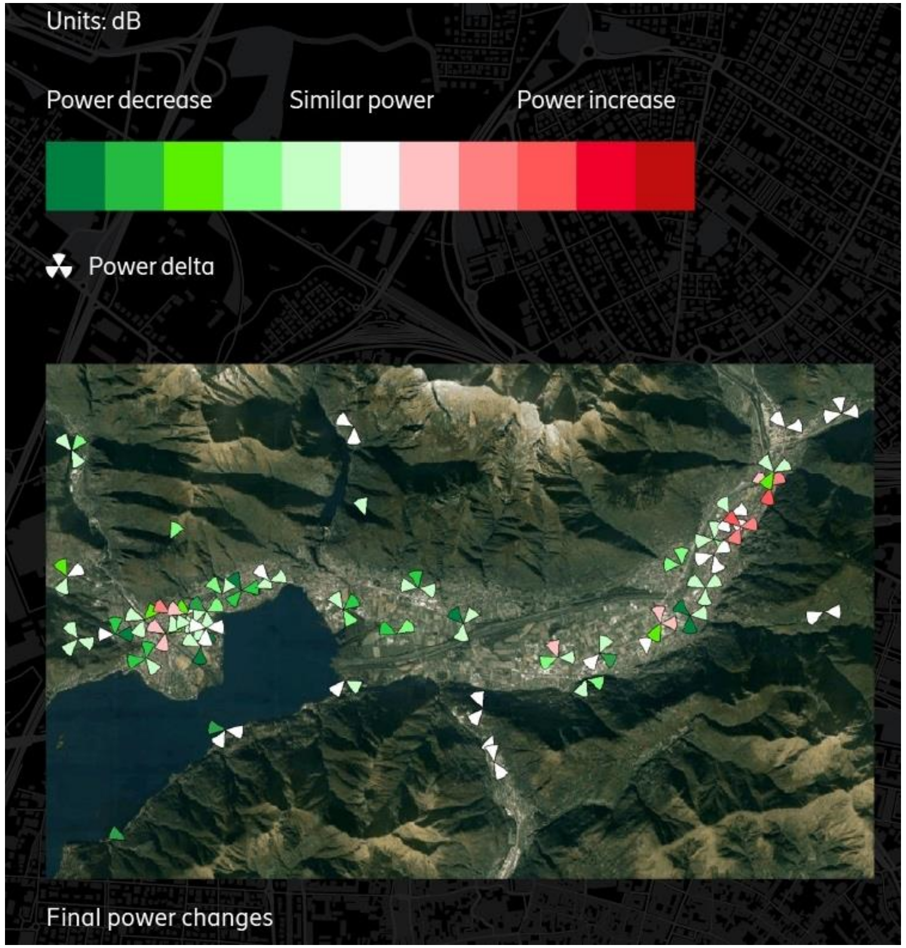
Reinforcement learning, some applications

- Design of clinical trials, Williamson (2019)
- Robot control, Kober (2013)



Reinforcement learning, some applications

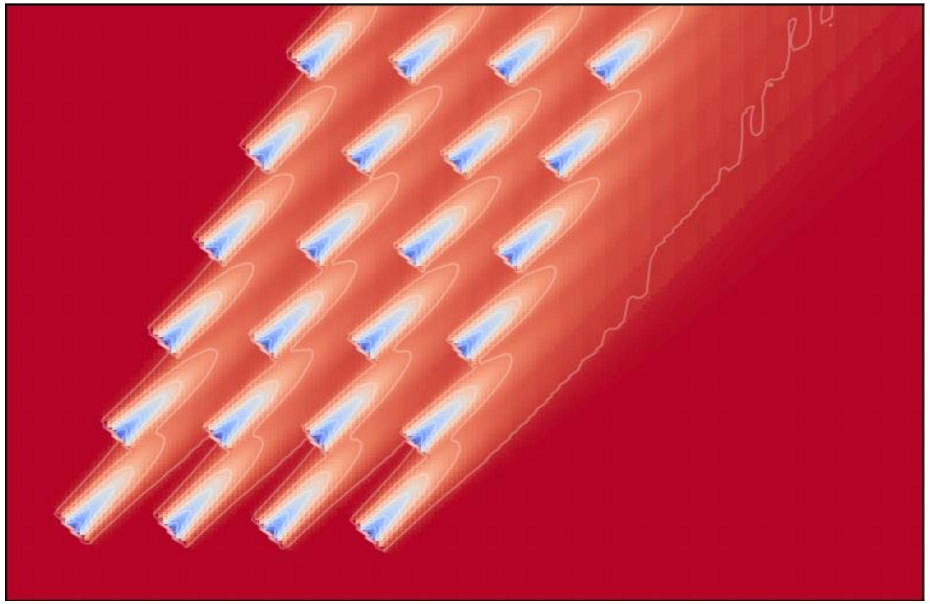
- Design of clinical trials, Williamson (2019)
- Robot control, Kober (2013)
- Telecom, *Ericsson* (2020)



Reinforcement learning, some applications

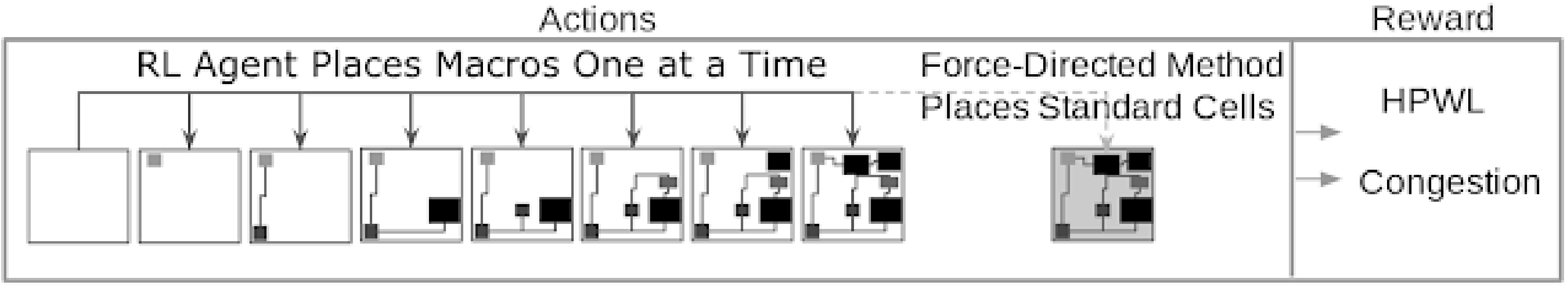
- Design of clinical trials, Williamson (2019)
- Robot control, Kober (2013)
- Telecom, *Ericsson* (2020)
- Wind farm control, Verstraeten (2020)

Simulator



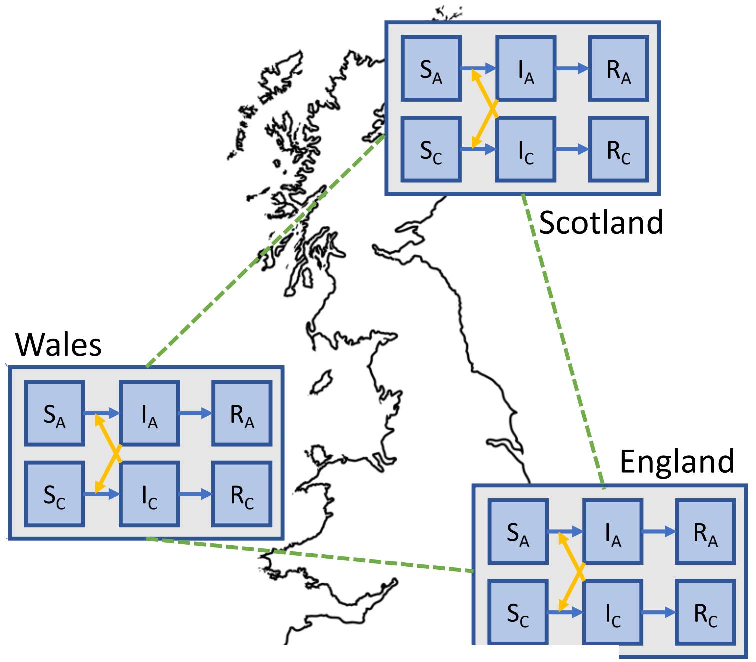
Reinforcement learning, some applications

- Design of clinical trials, Williamson (2019)
- Robot control, Kober (2013)
- Telecom, *Ericsson* (2020)
- Wind farm control, Verstraeten (2020)
- Chip design, Mirhoseini (2020)



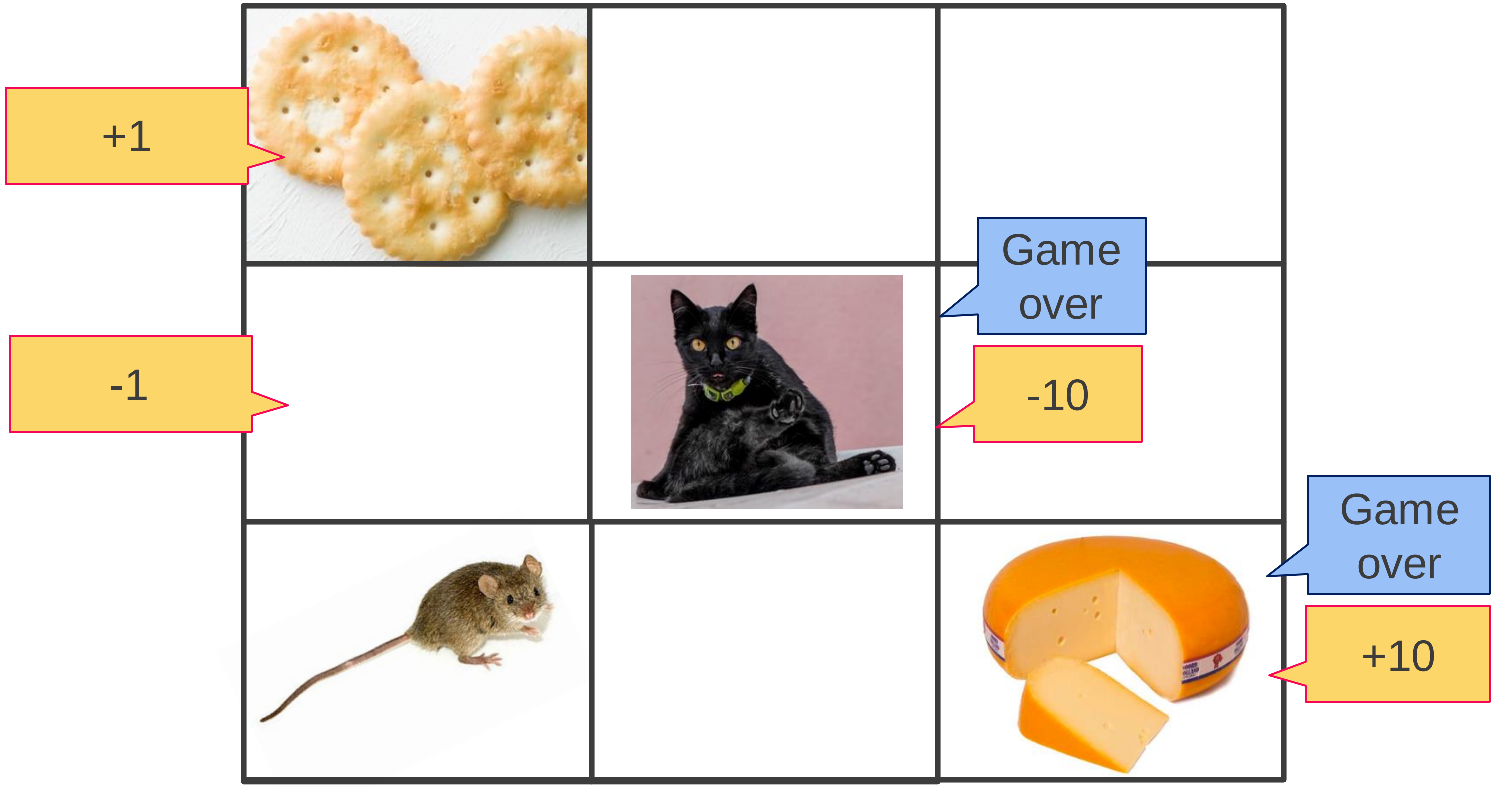
Reinforcement learning, some applications

- Design of clinical trials, Williamson (2019)
- Robot control, Kober (2013)
- Telecom, *Ericsson* (2020)
- Wind farm control, Verstraeten (2020)
- Chip design, Mirhoseini (2020)
- Epidemic control, Libin (2020)



The reinforcement learning problem

- Reinforcement learning:
  - We have an **agent** that wants to learn an optimal **policy**, by inter**acting** with an **environment,** by maximising a **reward** signal

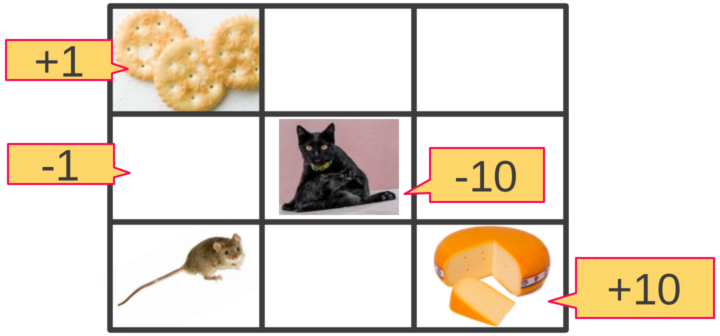


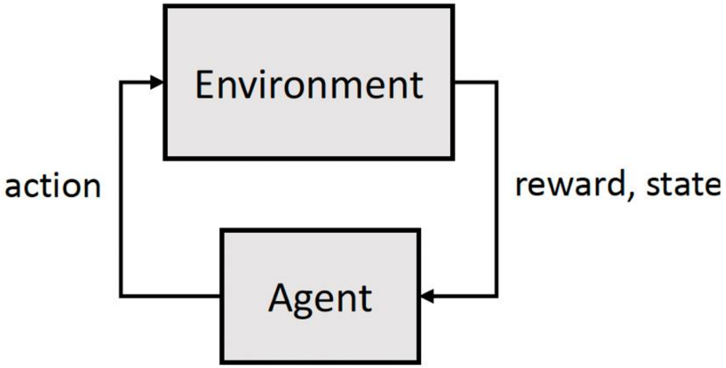
Markov decision process

Markovian

A Markov Decision Process corresponds to a tuple $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, where

- $\mathcal{S}$ is the set of possible states the environment can take upon
- $\mathcal{A}$ is the set of possible actions an agent can take
- $T\left(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}\right)$ signifies the transition probability to go from state $\mathbf{s}$ to state $\mathbf{s}'$ by taking an action $\mathbf{a}$
- γ is the discount factor that modulates the importance of future rewards
- $R\left(\mathbf{s}, \mathbf{a}, \mathbf{s}'\right)$ is the reward function that specifies which reward the agents receives upon choosing an action $\mathbf{a}$ in state $\mathbf{s}$





Policy

A Markov Decision Process corresponds to a tuple $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, where

- $\mathcal{S}$ is the set of possible states the environment can take upon
- $\mathcal{A}$ is the set of possible actions an agent can take
- $T\left(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}\right)$ signifies the transition probability to go from state $\mathbf{s}$ to state $\mathbf{s}'$ by taking an action $\mathbf{a}$
- γ is the discount factor that modulates the importance of future rewards
- $R\left(\mathbf{s}, \mathbf{a}, \mathbf{s}'\right)$ is the reward function that specifies which reward the agents receives upon choosing an action $\mathbf{a}$ in state $\mathbf{s}$

Given a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, an agent follows a policy p, that expresses the probability to take action $\mathbf{a}$ when in state $\mathbf{s}$:

$$p : \mathcal{S} \times \mathcal{A} \to [0, 1]$$

Policy - Cumulative reward

Return

One trajectory through the MDP

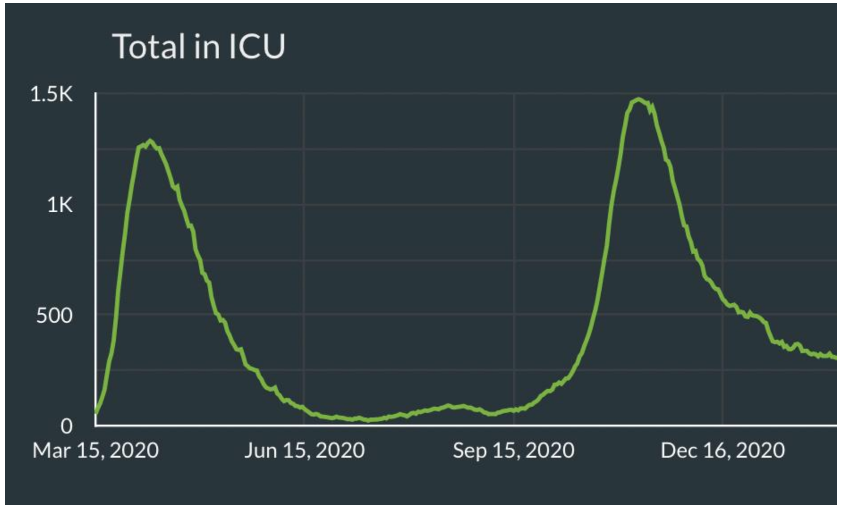
- In the MDP, we aim to maximize **cumulative reward**

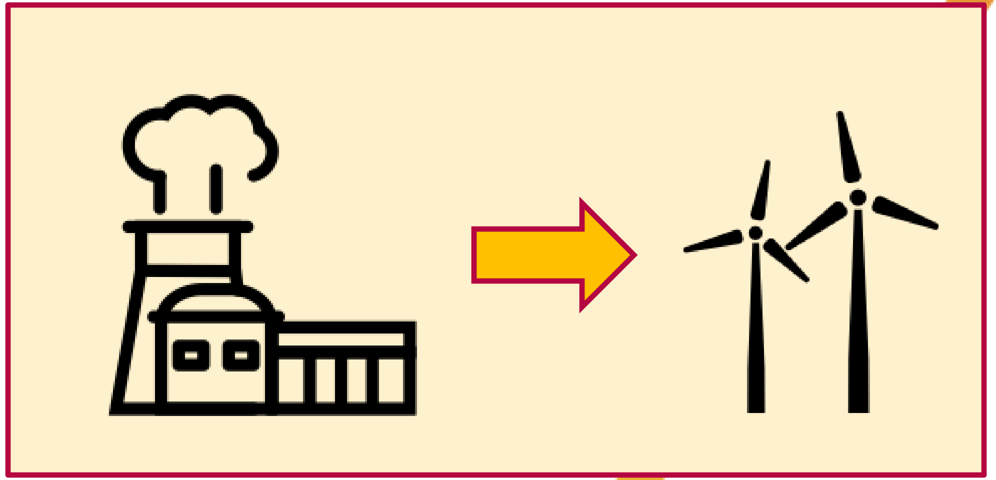
The return, or the discounted sum of rewards, starting from time step t, is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,,$$

where γ is the discount factor.

$r_{t+1} = R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$

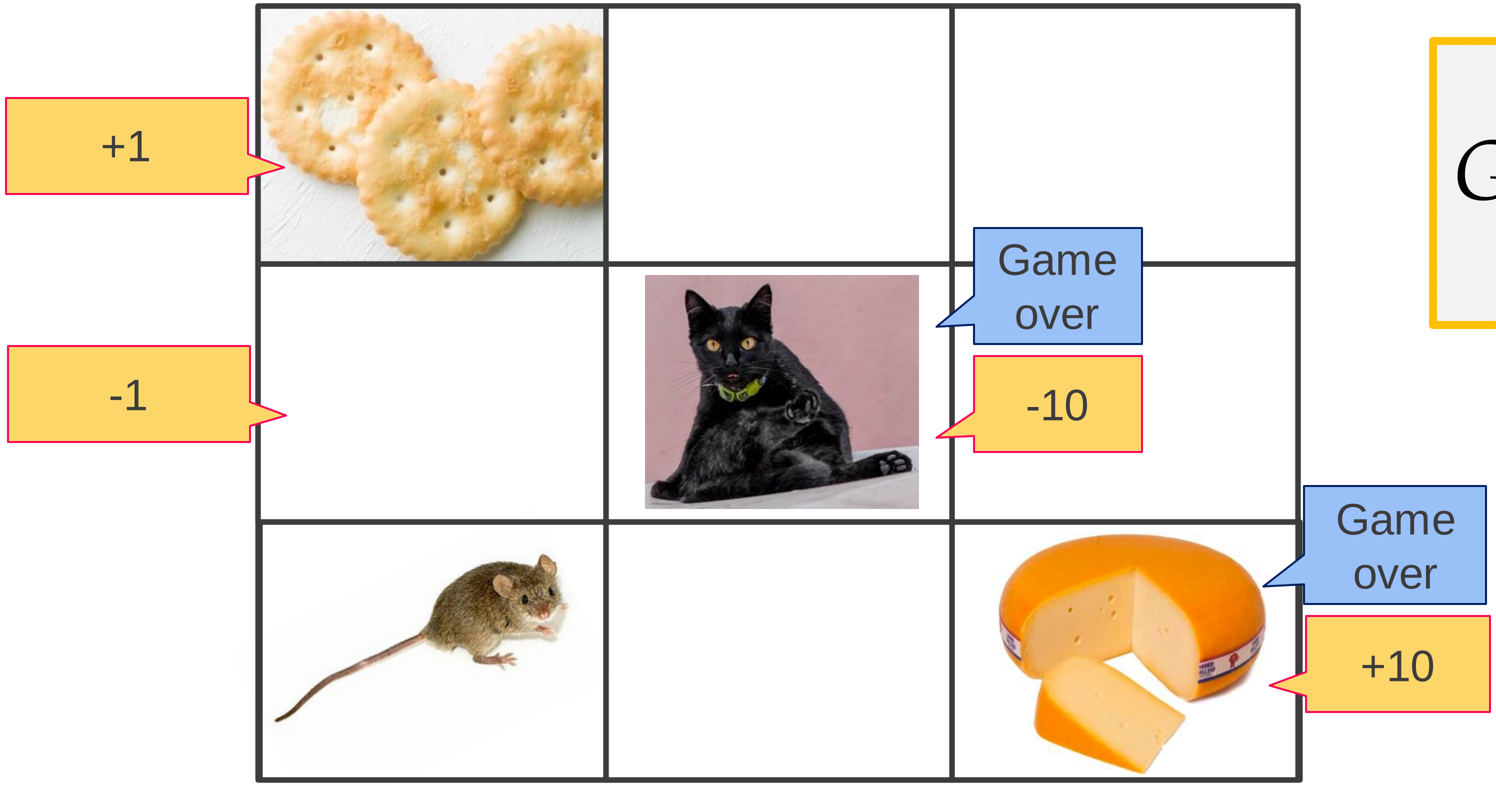




Policy

Given a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, an agent follows a policy p, that expresses the probability to take action a when in state s:

$$p : \mathcal{S} \times \mathcal{A} \to [0, 1]$$

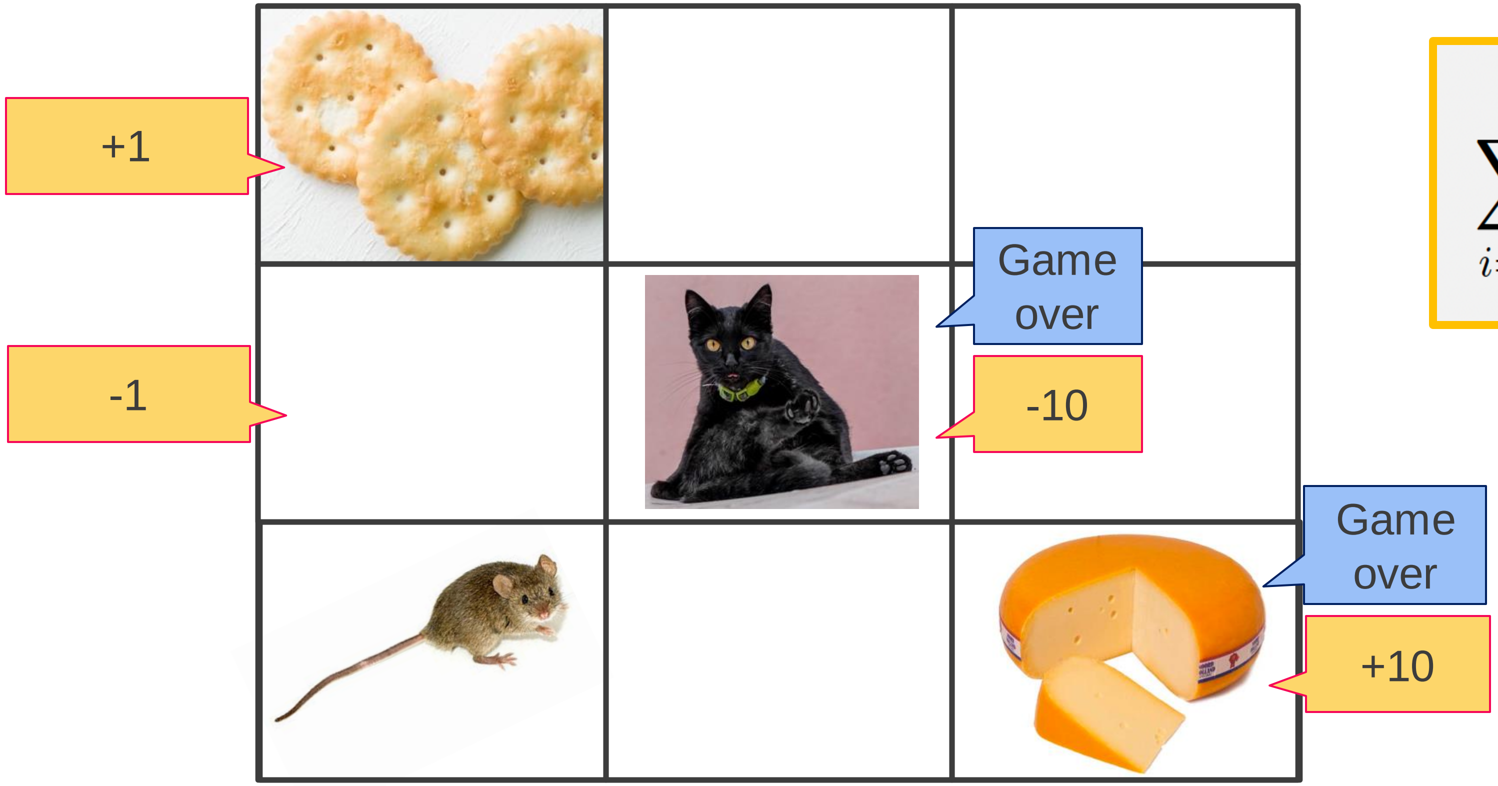


$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Policy

Given a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, an agent follows a policy p, that expresses the probability to take action $\mathbf{a}$ when in state $\mathbf{s}$:

$$p : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$$

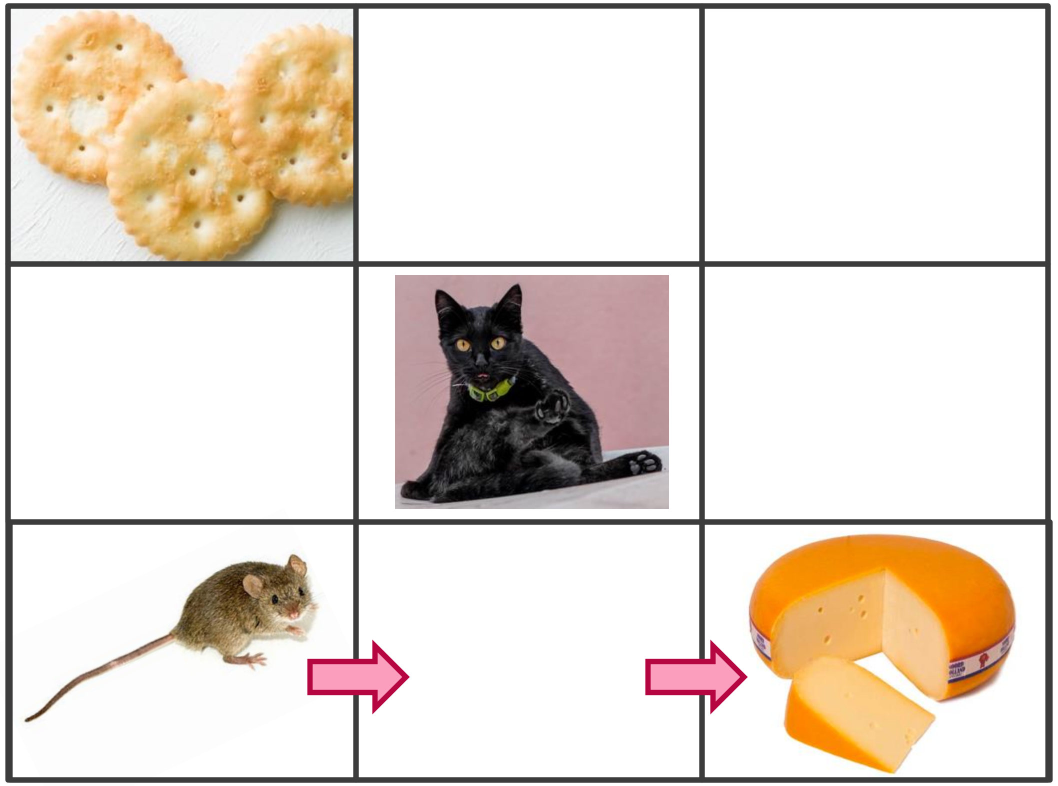


$$\sum_{i=0}^{T} d^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right)$$

Policy

Given a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, an agent follows a policy p, that expresses the probability to take action $\mathbf{a}$ when in state $\mathbf{s}$:

$$p : \mathcal{S} \times \mathcal{A} \to [0, 1]$$

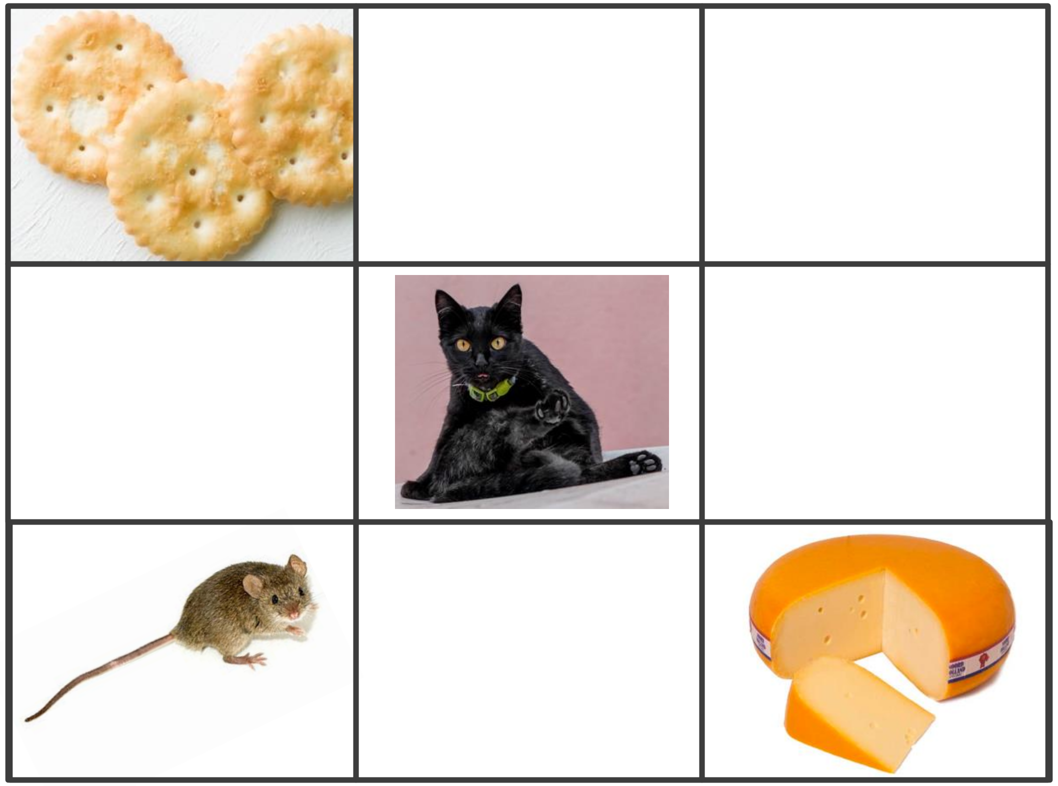


$p(.,\mathrm{F}) = 1$

Policy

Given a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, an agent follows a policy p, that expresses the probability to take action $\mathbf{a}$ when in state $\mathbf{s}$:

$$p : \mathcal{S} \times \mathcal{A} \to [0, 1]$$

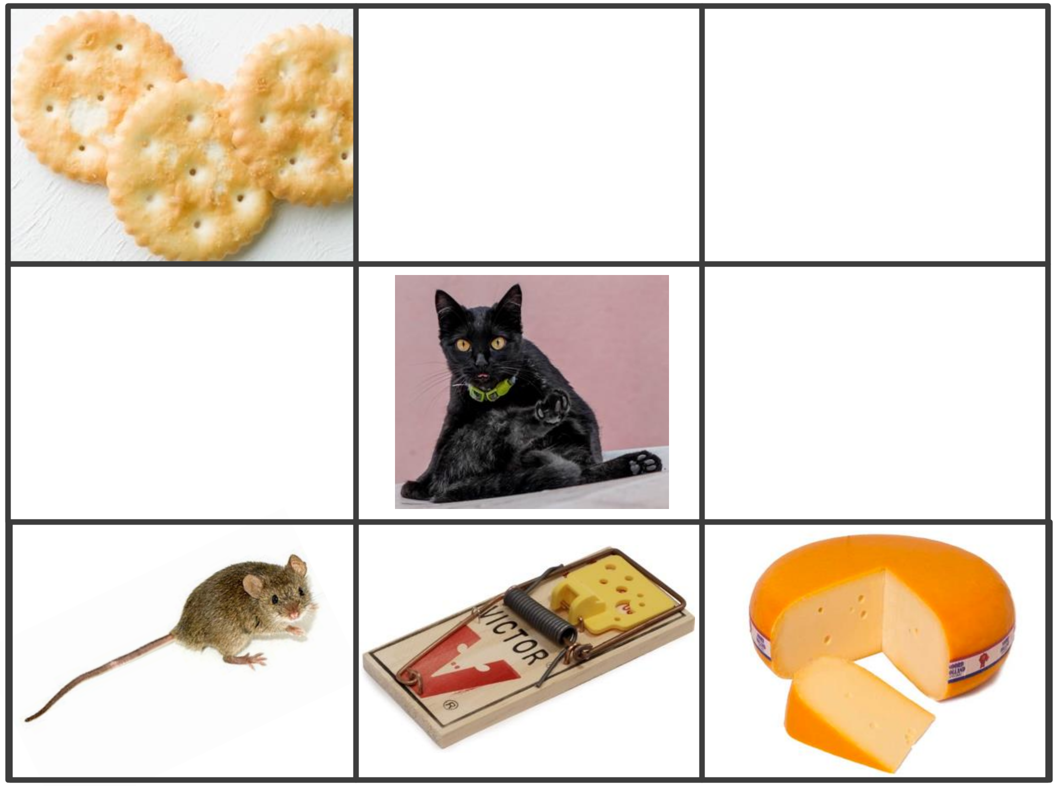


$p(.,.)$ = “Avoid cat”

Policy

Given a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, \Upsilon, R \rangle$, an agent follows a policy p, that expresses the probability to take action a when in state s:

$$p : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$$



$p(.,.)$ = “Avoid cat”

Policies, some intuition

- What is a policy to you?

- How to treat a patient that is admitted to the hospital?
- How to conduct a lab experiment?
- How to collect data?

Value functions

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- How good is it to be in a particular state

We define the value of being in state $\mathbf{s}$, following a policy p, as:

$$V^p(\mathbf{s}) = \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}]$$

Value functions

$$\sum_{i=0}^{T} d^i R(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1})$$

- How good is it to be in a particular state

We define the value of being in state $\mathbf{s}$, following a policy p, as:

$$V^p(\mathbf{s}) = \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}]$$

- How good is it to pick an action, when in a particular state

We define the value of being in state $\mathbf{s}$, when taking action $\mathbf{a}$, while following a policy p, as:

$$Q^p(\mathbf{s}, \mathbf{a}) = \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}]$$

Value function

- <u>Observation</u>:
  - from an **optimal Q function**, we can derive an optimal policy

$Q^*(s,a)$

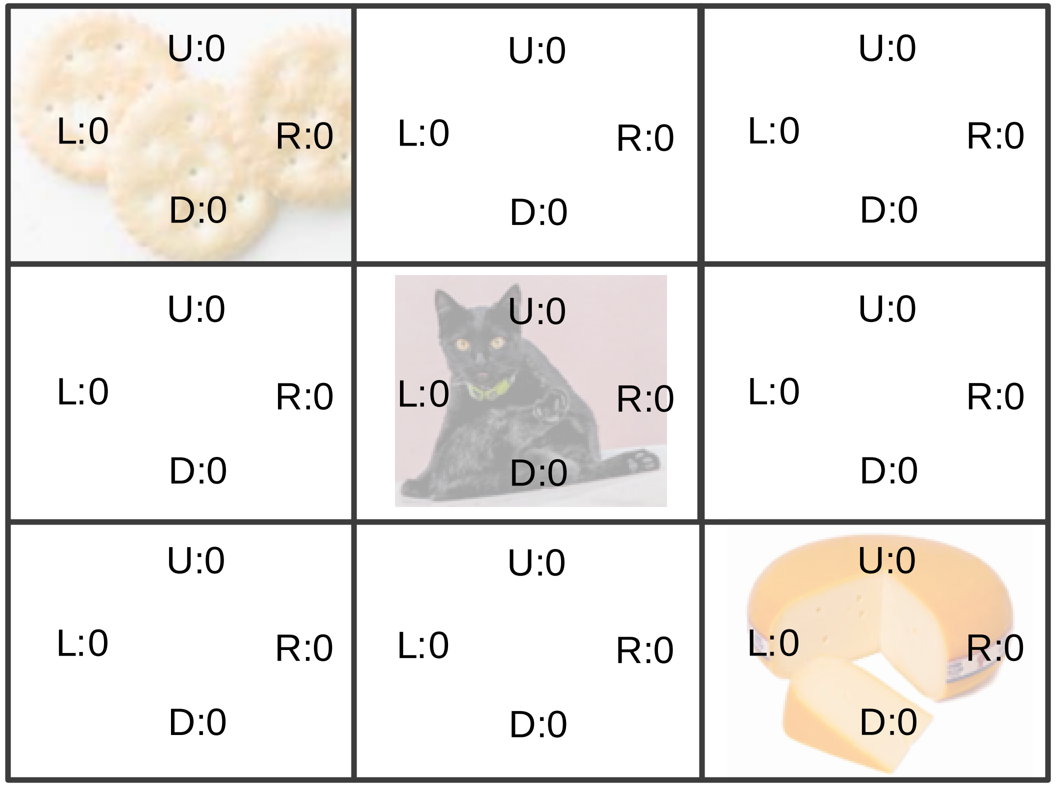
- Try to estimate $Q^*(s,a)$:
  - Q-learning

Bellman equation

$$Q^*(\mathbf{s},\mathbf{a}) = \mathbb{E}[r_{t+1} + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}_{t+1},\mathbf{a}')|\mathbf{s}_t = \mathbf{s},\mathbf{a}_t = \mathbf{a}]$$

Q-learning

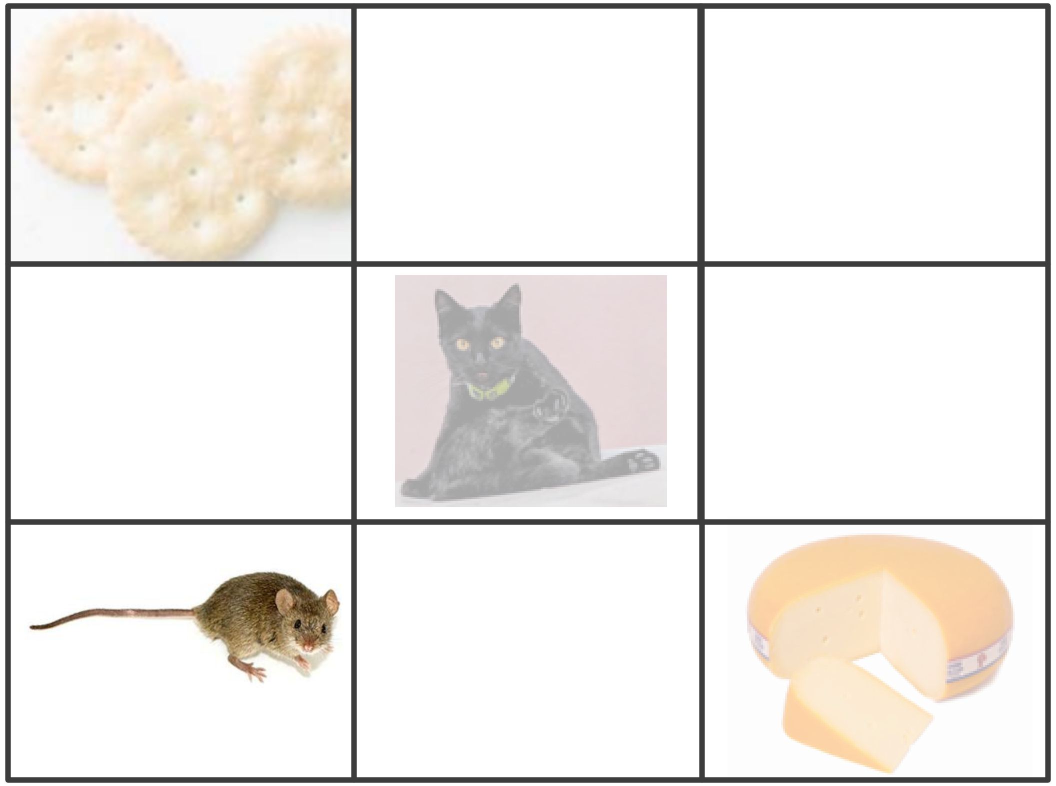
- Estimate Q*, by maintaining a Q-table:



Q-learning

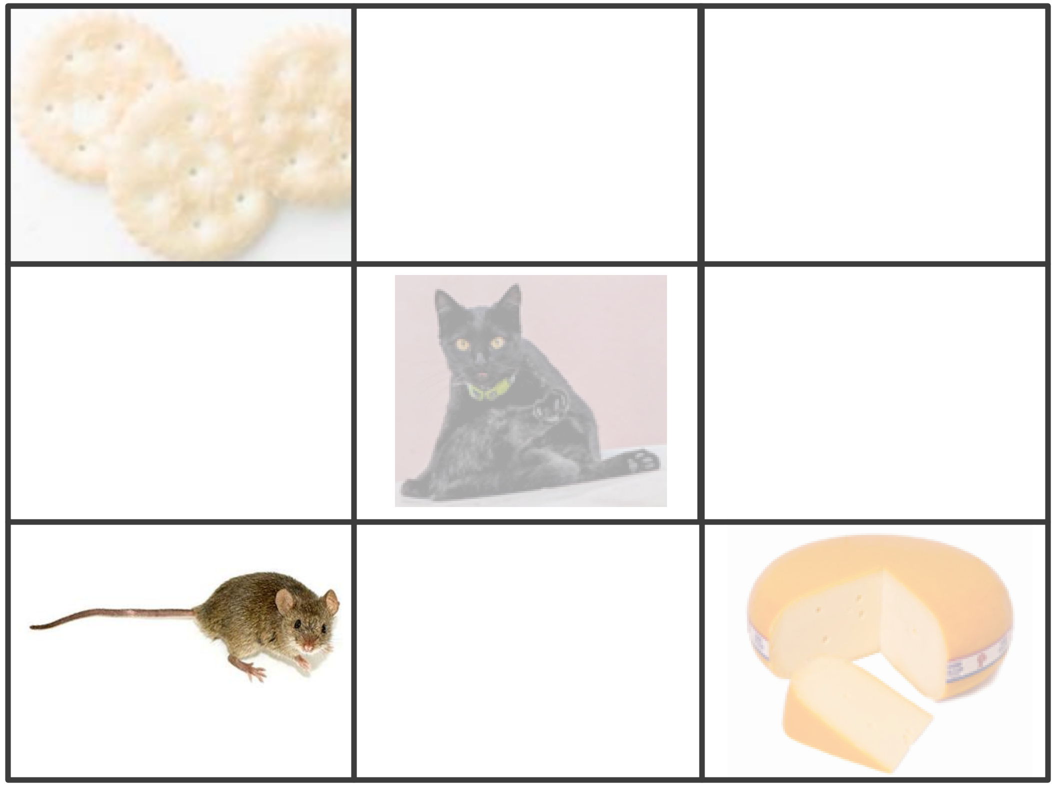
- Estimate Q*, by maintaining a Q-table:

Episodes



Q-learning

- Estimate Q*, by maintaining a Q-table:



Exploit the best values

At the start,
all values are zero

We need:
exploration

Exploitation vs Exploration

- Crucial tradeoff to enable reinforcement learning

- Experiment:
  - What was the first beer you drank?
  - Is your current favorite the same as your first beer?

Q-learning

Watkins, 1999

Given: a learning rate α

$\forall \mathrm{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$: initialize $\hat{Q}(\mathbf{s}, \mathbf{a})$

for *each episode* e **do**

  Initialize s_0

  for *each step in* e: $i = 0, 1, \ \ldots$ **do**

    Choose $\mathbf{a}_i$ from $\mathbf{s}_i$ using a policy derived from $\hat{Q}(\mathbf{s}, \mathbf{a})$

    Take action $\mathbf{a}_i$ and observe reward $\mathbf{r}_{i+1}$ and state $\mathbf{s}_{i+1}$

    $\hat{Q}(\mathbf{s}_i, \mathbf{a}_i) \leftarrow \hat{Q}(\mathbf{s}_i, \mathbf{a}_i) + \alpha \left[\mathbf{r}_{i+1} + \gamma \max_{\mathbf{a}'} \hat{Q}(\mathbf{s}_{i+1}, \mathbf{a}') - \hat{Q}(\mathbf{s}_i, \mathbf{a}_i) \right]$

    $\mathbf{s}_i \leftarrow \mathbf{s}_{i+1}$

  end

end

To balance exploration and exploitation

$1 - \epsilon$: $\max_{\mathbf{a}} \hat{Q}(\mathbf{s}, \mathbf{a})$

ϵ : random action

Q-learning

Convergence guarantees

TD-learning

Given: a learning rate α

$\forall \mathrm{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$: initialize $\hat{Q}(\mathbf{s}, \mathbf{a})$

for *each episode* e **do**

Initialize s_0

for *each step in* e: $i = 0, 1, \ \ldots$ **do**

Choose $\mathbf{a}_i$ from $\mathbf{s}_i$ using a policy derived from $\hat{Q}(\mathbf{s}, \mathbf{a})$

Take action $\mathbf{a}_i$ and observe reward $\mathbf{r}_{i+1}$ and state $\mathbf{s}_{i+1}$

$$\hat{Q}(\mathbf{s}_i, \mathbf{a}_i) \leftarrow \hat{Q}(\mathbf{s}_i, \mathbf{a}_i) + \alpha \left[\overbrace{\underbrace{\mathbf{r}_{i+1} + \gamma \max_{\mathbf{a}'} \hat{Q}(\mathbf{s}_{i+1}, \mathbf{a}')}_{\text{new value}} - \underbrace{\hat{Q}(\mathbf{s}_i, \mathbf{a}_i)}_{\text{old value}}}^{\text{temporal difference}} \right]$$

$\mathbf{s}_i \leftarrow \mathbf{s}_{i+1}$

end

end

Reinforcement learning, some milestones

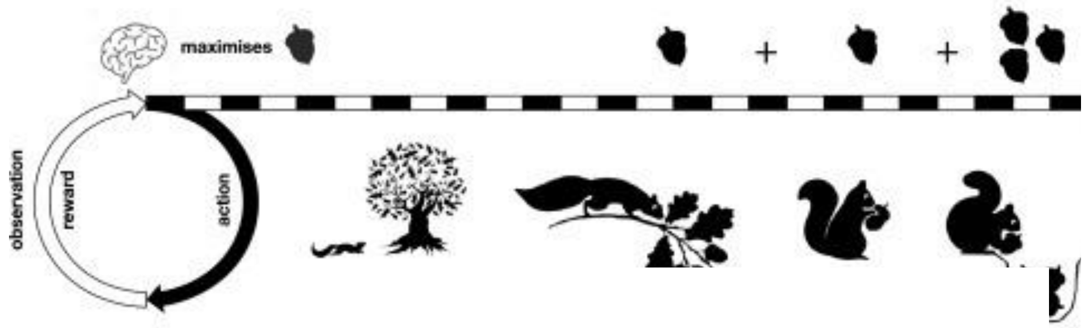
- Backgammon RL agent, Tesauro (1992)
- Reinforcement Learning book, Sutton and Barto (1998)
- Atari agent, *Deepmind* (2015)

Reinforcement learning, some milestones

- Backgammon RL agent, Tesauro (1992)
- Reinforcement Learning book, Sutton and Barto (1998)
- Atari agent, *Deepmind* (2015)
- AlphaGo, *Deepmind* (2016)

Reinforcement learning, some milestones

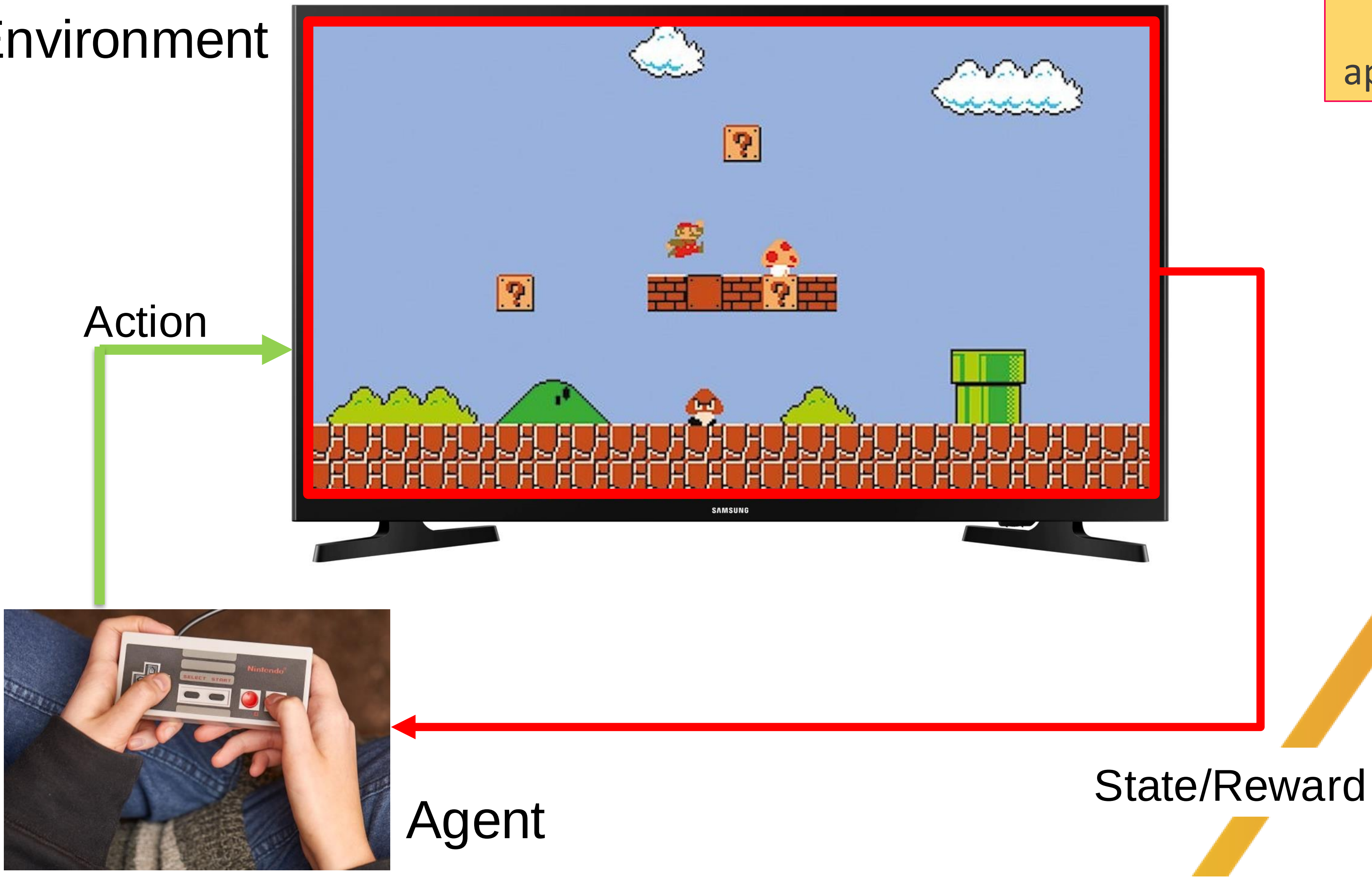
- Backgammon RL agent, Tesauro (1992)
- Reinforcement Learning book, Sutton and Barto (1998)
- Atari agent, *Deepmind* (2015)
- AlphaGo, *Deepmind* (2016)
- AlphaGo Zero, *Deepmind* (2017)
- OpenAI Five, *OpenAI* (2018)
- “Reward is enough”, Silver et al. (2021)



Q-learning in a large state space

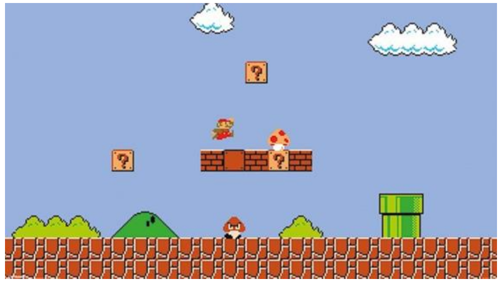
$$Q\left(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta}\right)$$

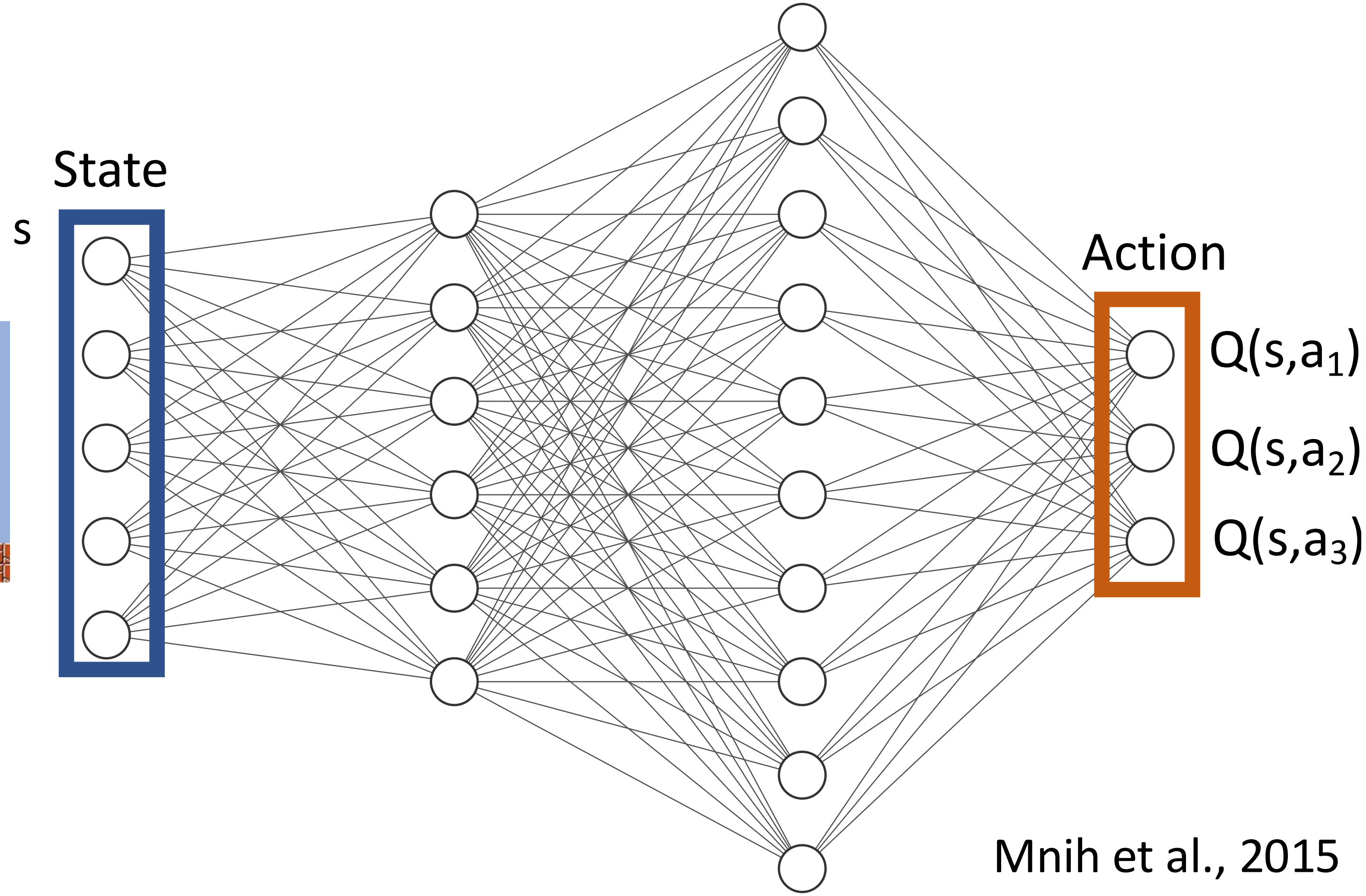
Function approximation



Deep Q-networks (DQN)

$$Q\left(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta}\right)$$





Mnih et al., 2015

Deep Q-networks

$$\boxed{Q\left(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta}\right)}$$

- Minimize temporal difference error, from mini-batch:

$$\{\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}_{i+1}\}_{i=0}^{N}$$

- We could use the mean squared error:

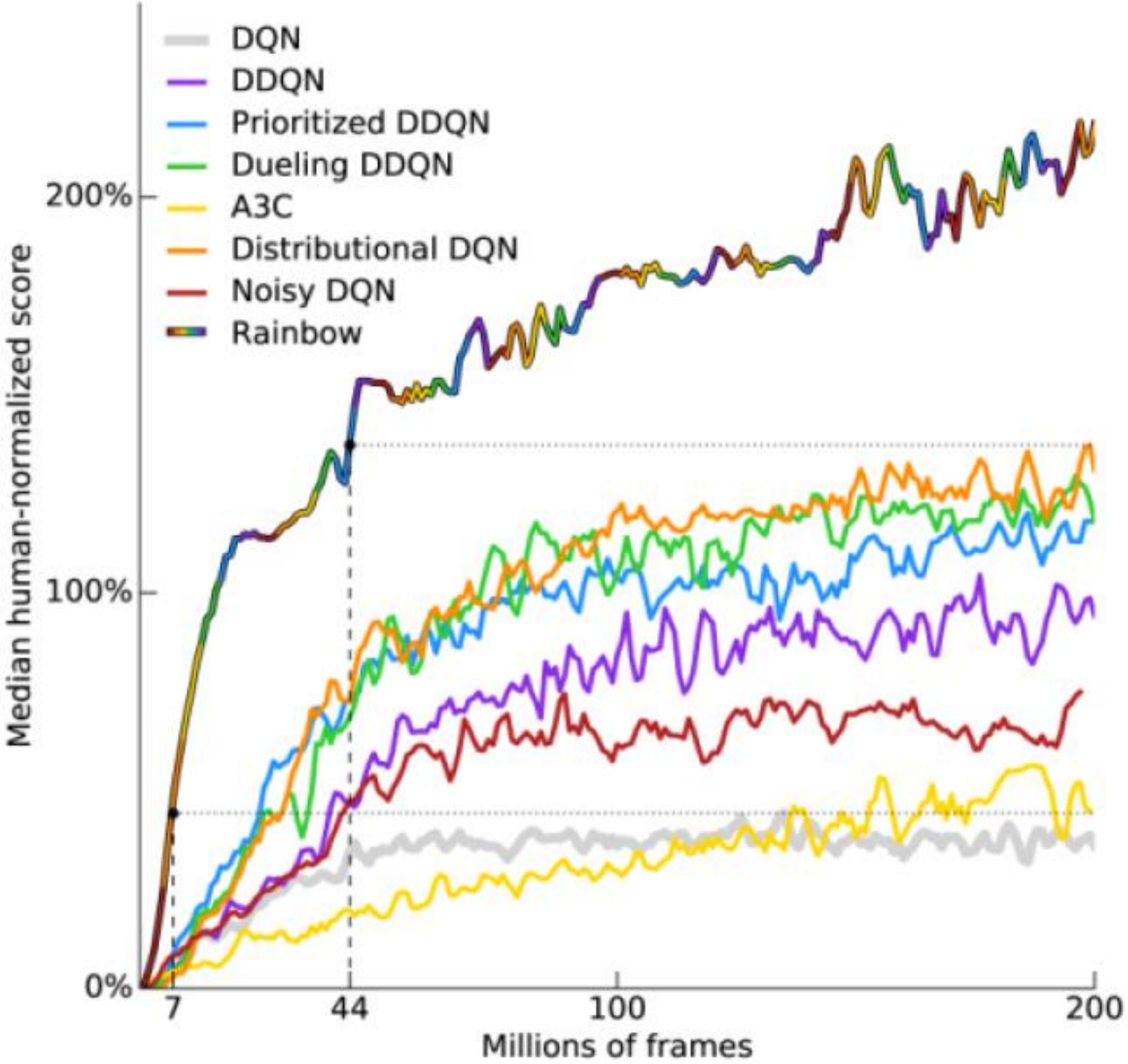
$$L\left(\mathbf{s}, \mathbf{a}\right) = \frac{1}{N} \sum_i \left(y_i - Q\left(\mathbf{s}_i, \mathbf{a}_i \mid \boldsymbol{\theta}\right)\right)^2$$, where,

$$y_i = \mathbf{r}_i + \gamma \max_{\mathbf{a}} Q\left(\mathbf{s}_{i+1}, \mathbf{a} \mid \phi\right)$$

Use a separate target network

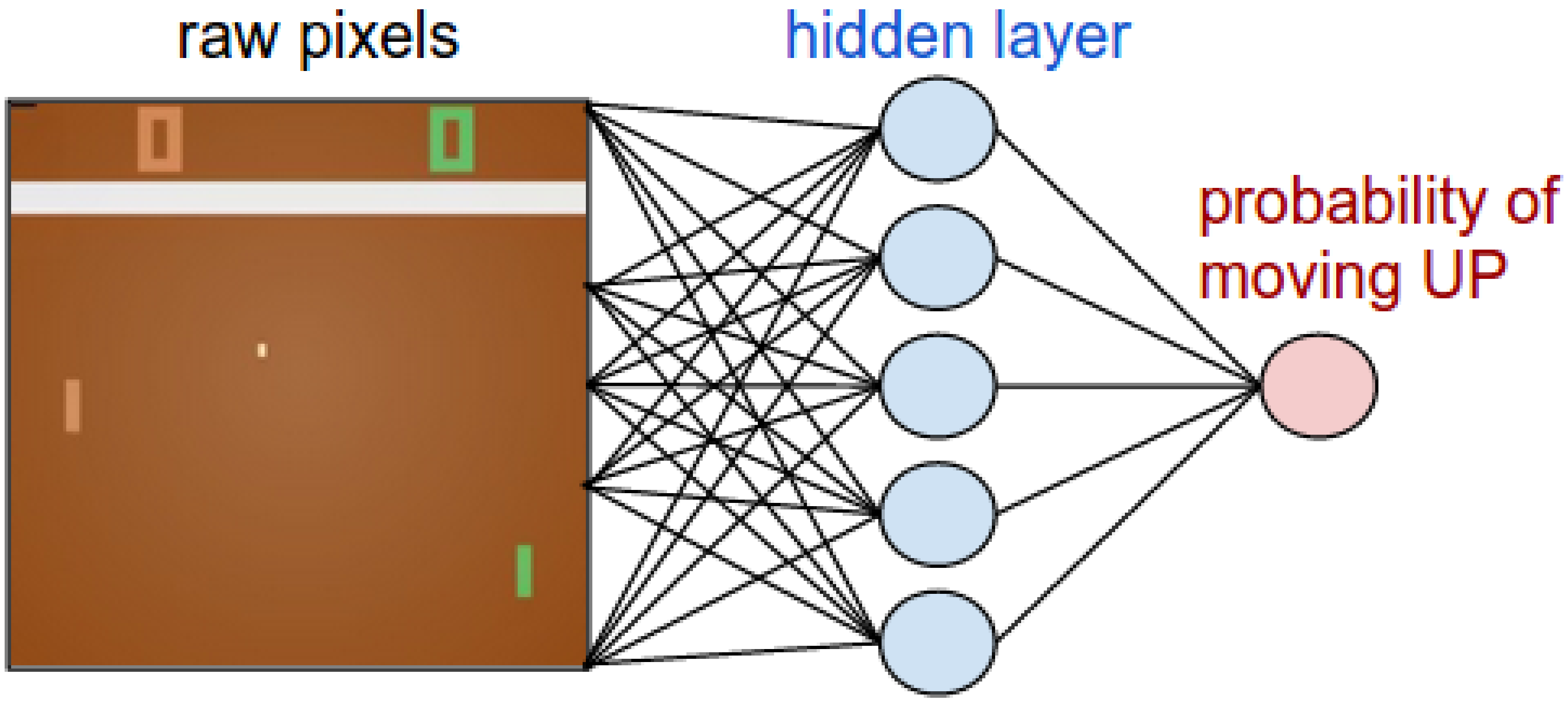
Unstable

Deep Q-networks



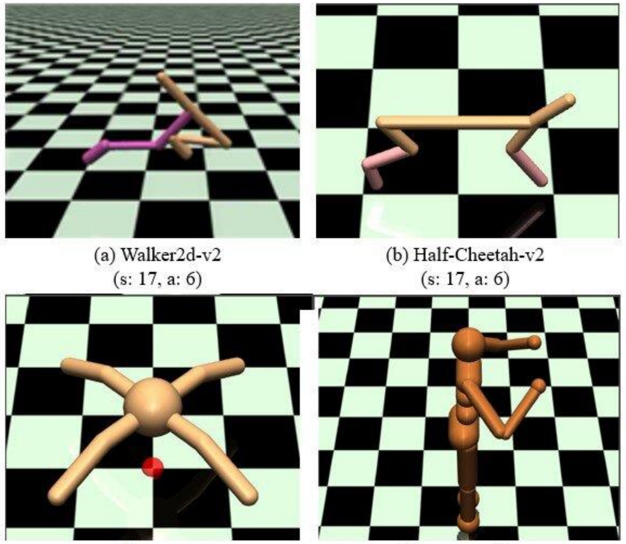
Hessel, 2017

Policy gradient RL methods



Policy gradient methods: advantages

- Simplicity: We can estimate the policy directly
- Stochastic policies:
  - We don’t need to handle the exploration/exploitation trade-off by hand
  - Effective in high-dimensional and/or continuous actions spaces



(a) Walker2d-v2 (s: 17, a: 6)
(b) Half-Cheetah-v2 (s: 17, a: 6)
(c) Ant-v2 (s: 111, a: 8)
(d) Humanoid-v2 (s: 376, a: 17)

Guofei et al., 2023

Policy gradient methods: disadvantages

- Sample inefficient
- High variance

Policy gradient

- Stochastic policy:

$$p_\theta(s) = P(\mathcal{A}|s;\theta)$$

- Objective function:

$$G(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{k+1}$$

$\boldsymbol{\tau}$ is a trajectory:
$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \dots$

G($\boldsymbol{\tau}$) is the **return** of trajectory $\boldsymbol{\tau}$

Policy gradient

- Stochastic policy:

$$p_\theta(s) = P(\mathcal{A}|s;\theta)$$

- Objective function:

$$G(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{k+1}$$

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[G(\tau)]$$

Objective function

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[G(\tau)]$$

$$J(\theta) = \sum_{\tau} P(\tau|\theta) G(\tau)$$

$$\boxed{P(\tau|\theta) = \prod_{t=0} T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p_\theta(\mathbf{a}_t|\mathbf{s}_t)}$$

Maximise the objective function

$$\max_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[G(\tau)]$$

$$J(\theta) = \sum_{\tau} P(\tau|\theta) G(\tau)$$

$$P(\tau|\theta) = \prod_{t=0} T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

- Gradient ascent: $\theta = \theta + \alpha \nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[\nabla_\theta \log p_\theta(\mathbf{a}_t|\mathbf{s}_t) G(\tau)]$$

Policy gradient theorem

Sample based approximation

$$\boxed{G(\tau, t) = \sum_{k=t}^{\infty} \gamma^k r_{k+1}}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[\nabla_\theta \log p_\theta(\mathbf{a}_t|\mathbf{s}_t) G(\tau)]$$

- Use the policy p_θ to collect M episodes: $\tau_1, \ldots, \tau_M$

$$\nabla_\theta J(\theta) \approx \hat{g} = \frac{1}{M} \sum_{m=0}^{M} \sum_{t=0}^{\infty} \underline{\nabla_\theta \log p_\theta(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) G(\tau^{(i)}, t)}$$

Vanilla Policy gradient

Given: a learning rate $\alpha(.)$
Initialize policy parameter $\boldsymbol{\theta}$ and value function V
for $i = 1, 2 \ \ldots$ **do**
  Collect a set of trajectories Y using the current policy $p_{\boldsymbol{\theta}_i}$
  Compute the policy gradient estimate $\hat{g}_i$
  Update the policy: $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha(i)\,\hat{g}_i$
end

Williams, 1992

Adding a baseline

- **Problem**: high variance in the **return**
- Use a function to **compensate the variance** but **does not change the expectation**
- **Advantage:** $\hat{A}_t = G(\tau, t) - V(\mathbf{s}_t)$

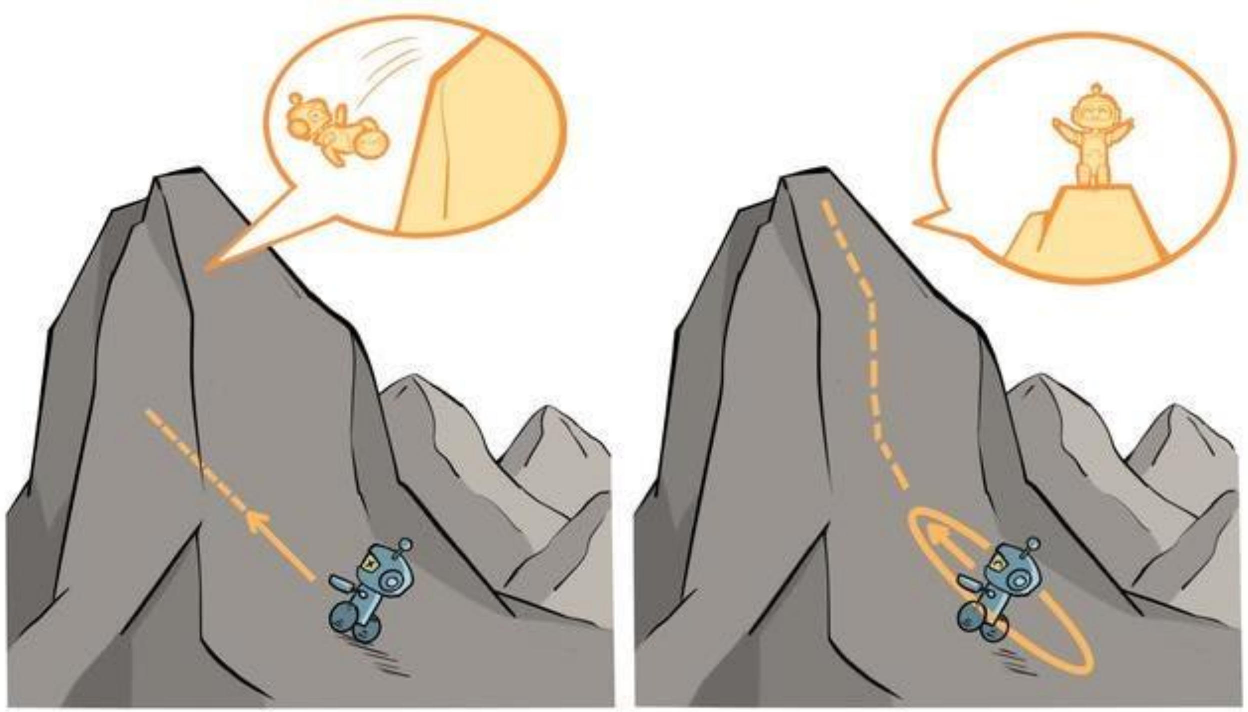
Policy gradient with an advantage

Given: a learning rate $\alpha(.)$
Initialize policy parameter $\boldsymbol{\theta}$ and value function V
for $i = 1, 2 \ \ldots$ **do**
  Collect a set of trajectories Y using the current policy $p_{\boldsymbol{\theta}_i}$
  for *each time step* t *and each trajectory* $y \in Y$ **do**
    Compute the advantage $\hat{A}_t$
  end
  Update $V \ (s_t)$
  Compute the policy gradient estimate $\hat{g}_i$
  Update the policy: $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha(i)\,\hat{g}_i$
end

Williams, 1992

Trust region policy optimisation

- **Problem**: Learning **instability**
- Gradient updates **change** the policy too **much**



medium.com/@masoumzadeh

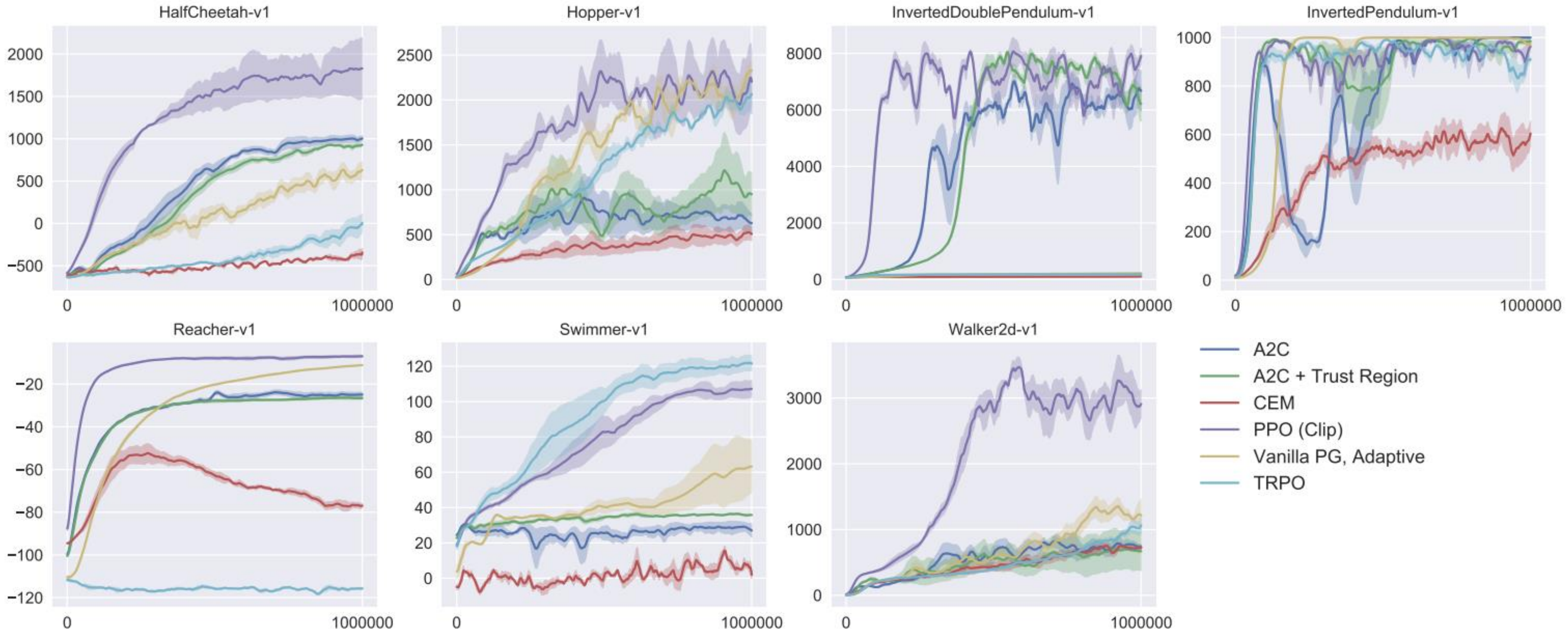
Trust region policy optimisation

- **Problem**: Learning **instability**
- Gradient updates **change** the policy too **much**
- Avoid moving away too much from the old policy:

$$D_{KL}(p_{\theta_{\text{old}}}(.|\mathbf{s})||p_\theta(.|\mathbf{s})) <= \delta$$

Proximal Policy Optimisation

PPO for the win

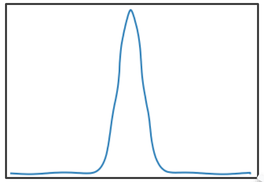


Multi-armed bandits

Stateless RL

Multi-armed bandits

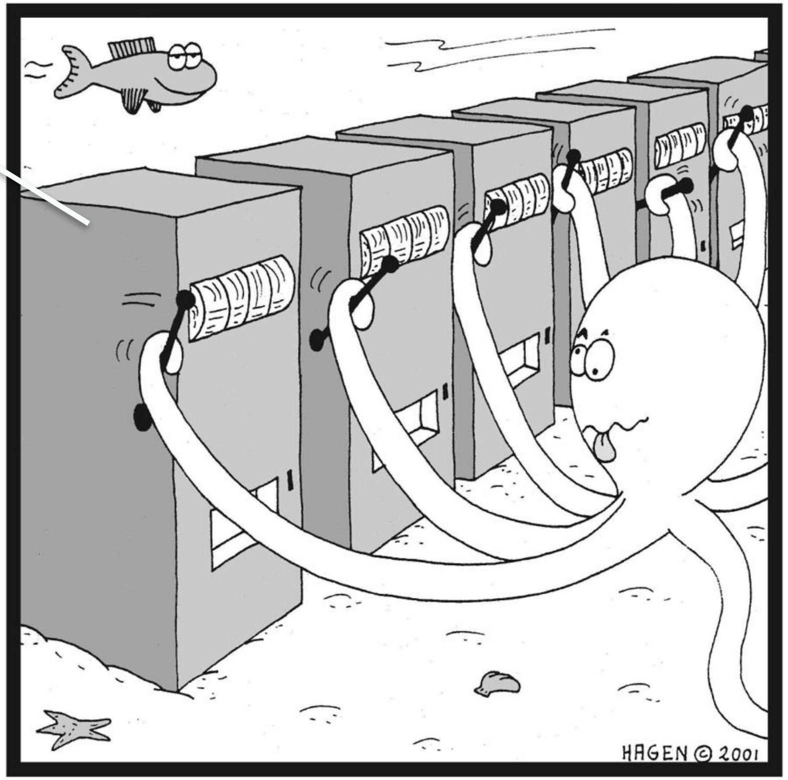
$\mathcal{R}_k$



Set of arms: $\{A_k\}_k^K$

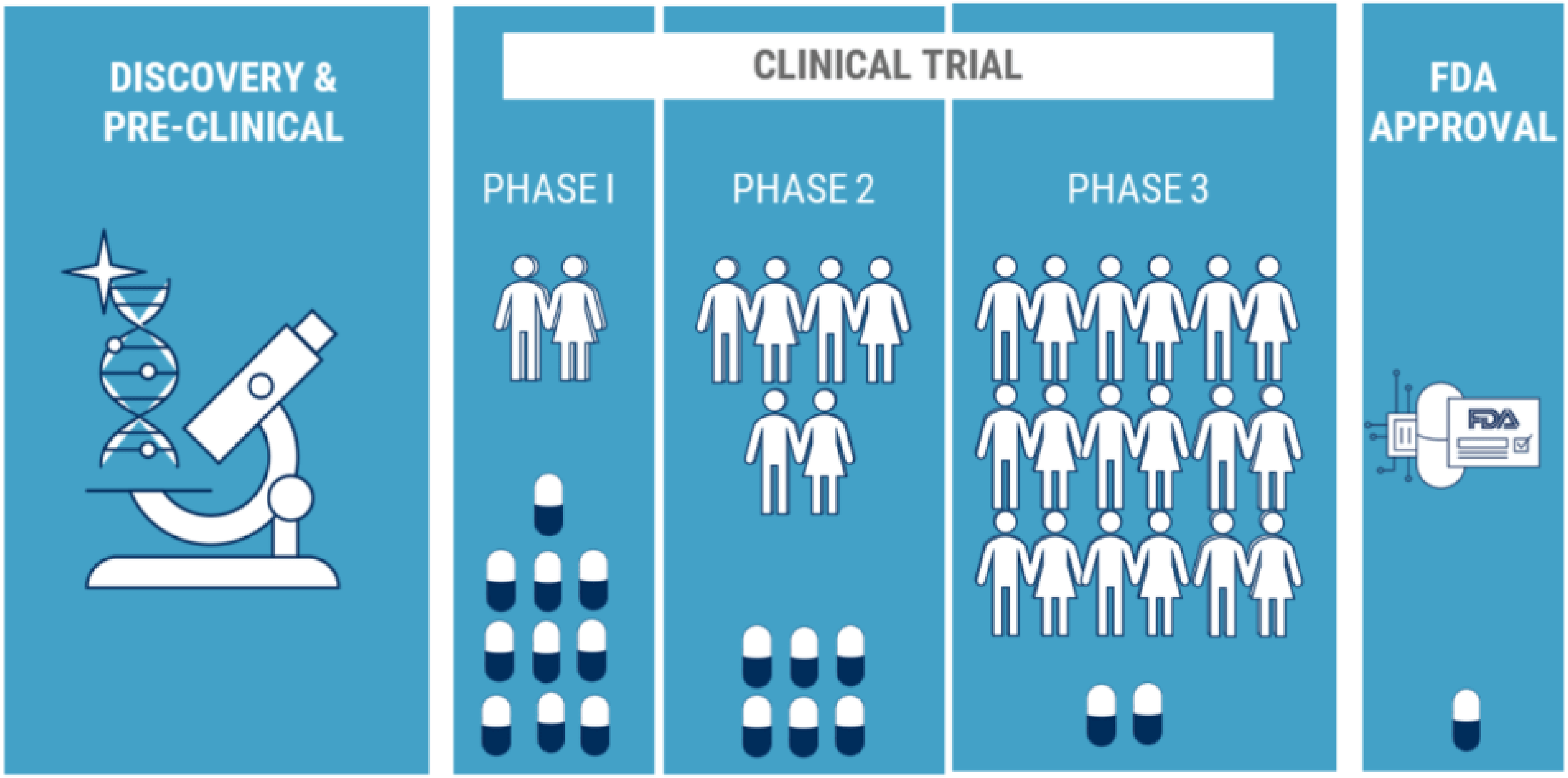
Stochastic reward: r_k

Arm's mean: $\mathbb{E}[r_k] = \mu_k$



Multi-armed bandit application

- Clinical trials



Cumulative regret

- ϵ - greedy
  - Exploit with probability $1 - \boxed{\epsilon}$

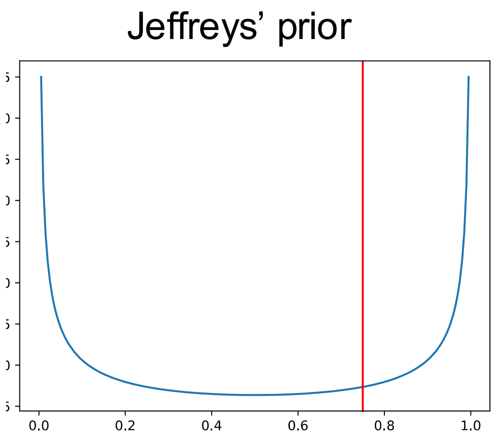
- UCB
  - $A_t = \text{argmax}_k \left[\hat{\mu}_k + \boxed{c} \sqrt{\frac{\ln(t)}{N_t(k)}} \right]$

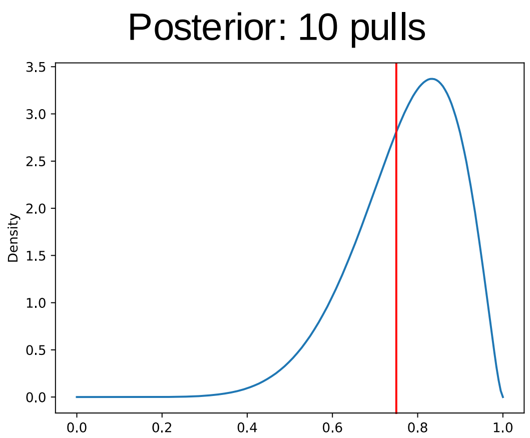
Use prior knowledge: Bayesian philosophy

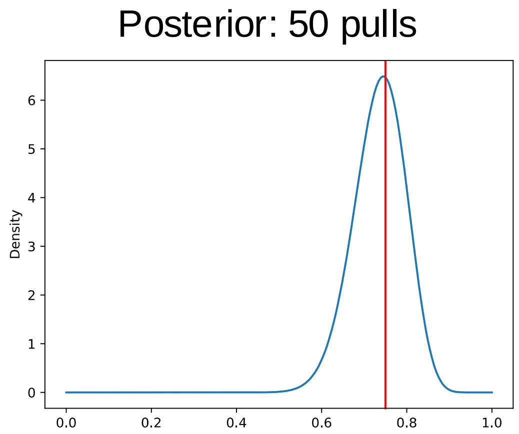
- It’s all about belief
  - Prior belief
  - Updating our belief, given data

Bayesian bandits

- Prior belief over means: $\pi(.)$
- History: $\mathcal{H}^{(t-1)} = \{a^{(i)}, r^{(i)}\}_{i=1}^{t-1}$
- Posterior: $\pi(\cdot \mid \mathcal{H}^{(t-1)})$







Cumulative regret: Thompson sampling

$$
\begin{array}{l}
\textbf{Given: } \pi(.) \text{ and } \mathcal{H}^{(0)} = \emptyset \\
\textbf{for } \; t = 1, \; \ldots, +\infty \; \textbf{do} \\
\quad \theta^{(t)} \sim \pi(\cdot \mid \mathcal{H}_{t-1}) \\
\quad a^{(t)} = \sigma_1(\theta^{(t)}) \\
\quad r^{(t)} \leftarrow \text{Pull arm } a^{(t)} \\
\quad \mathcal{H}^{(t)} \leftarrow \mathcal{H}^{(t-1)} \cup \{a^{(t)}, r^{(t)}\} \\
\textbf{end}
\end{array}
$$

(Thompson, 1933; Chapelle, 2011)

Pure exploration

- Decision making
  - e.g., Simulation of prevention strategies in a compute-intensive model
- Best-arm identification
  - fixed budget

Top-two Thompson sampling

Given: $\pi(.)$ and $\mathcal{H}^{(0)} = \emptyset$

for $t = 1, \ldots, +\infty$ **do**

$\quad \theta^{(t)} \sim \pi(\cdot \mid \mathcal{H}_{t-1})$

$\quad \boxed{a^{(t)} = \sigma_1(\theta^{(t)})}$ Resample

$\quad r^{(t)} \leftarrow$ Pull arm $a^{(t)}$

$\quad \mathcal{H}^{(t)} \leftarrow \mathcal{H}^{(t-1)} \cup \{a^{(t)}, r^{(t)}\}$

end

(Russo, 2016)

Things we did *not* talk about

- Mixing planning and learning
- Multi-objective RL
- Multi-agent RL
- Safety / Fairness / Trustworthiness
- Partially observable MDPs

Reading material

- RL book by Sutton and Barto
  - Mixing planning and learning
- Background and code
  - https://spinningup.openai.com/

Reading material

- DQN paper
  - Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.
- Policy gradient
  - Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In *ICML* (pp. 1889-1897).
- AlphaGo paper
  - Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.

Reading material

- Multi-objective RL:
  - Roijers, Diederik M., et al. "A survey of multi-objective sequential decision-making." *Journal of Artificial Intelligence Research* 48 (2013): 67-113.
- Multi-agent RL:
  - Nowé, Ann, Peter Vrancx, and Yann-Michaël De Hauwere. "Game theory and multi-agent reinforcement learning." *Reinforcement Learning*, 2012. 441-470.

Graphic credits

- Cats/dogs/crackers/drugs/ICU/bike/robot/LHC/beer/westvleteren/Las Vegas: unsplash
- Mouse/mouse trap/Atari/Bayes: Wikipedia
- Cheese: Kaasdok.nl
- Go board game: https://www.go-jigs.eu/what-is-go/
- Lee Sedol: New Yorker
- Clinical trials: https://www.cbinsights.com/
- Telecom application: Ericson website
- ICU covid: sciensano

Acknowledgments

- Youtube Videos on DQN by Olivier Sigaud
- Reinforcement learning (book), Sutton and Barto, 2020
- Introduction to bandits: adapted slides from mmds.org
- Policy gradient introduction: https://huggingface.co/

Thank you! Any more questions?

ai.vub.ac.be
@aibrussels

Pieter.Libin@vub.be
@PieterLibin