Introduction

Pieter.Libin@vub.be
@PieterLibin

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

ai.vub.ac.be
@aibrussels

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

Program

- 10h00-12h00: Lecture on Reinforcement Learning
- 13h00-15h00: RL Programming exercise in Python

- **Interactive session**: ask questions + participate!

Attendance

Course objectives

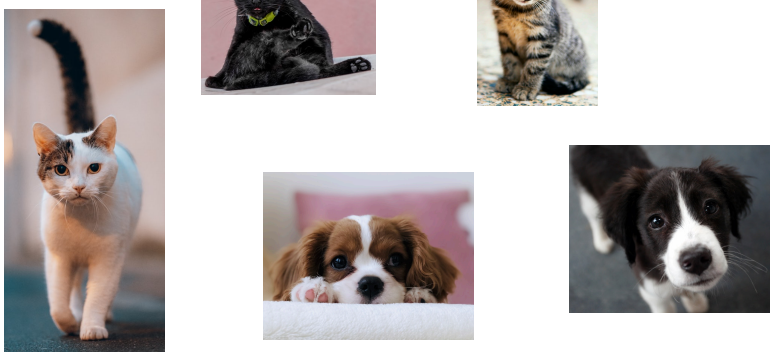
- **Intuition** about reinforcement learning
- How can *you* use reinforcement learning
- **Understanding** important reinforcement learning **algorithms**

Reinforcement learning, a bit different

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

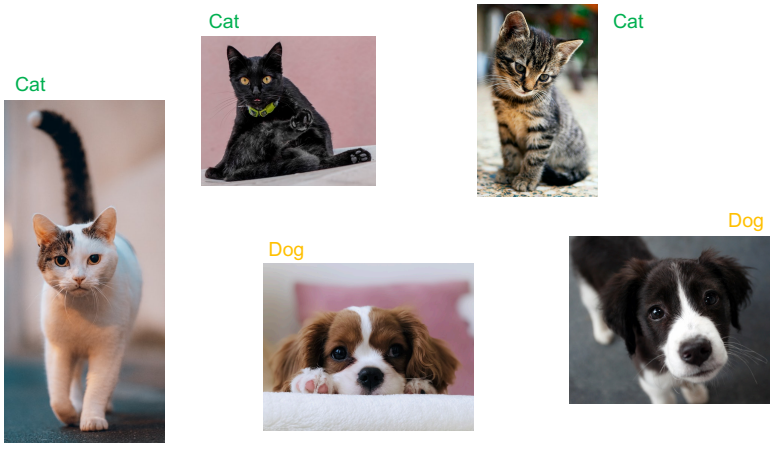
Reinforcement learning, a bit different

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



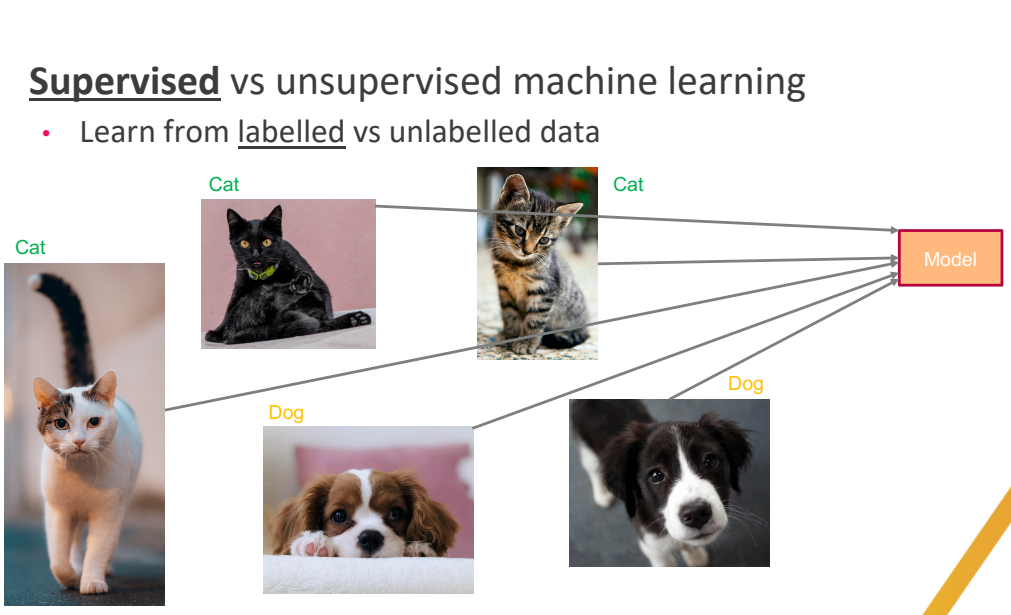
Reinforcement learning, a bit different

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



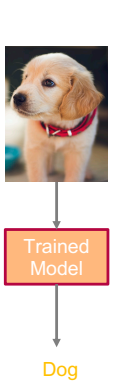
Reinforcement learning, a bit different

- **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

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

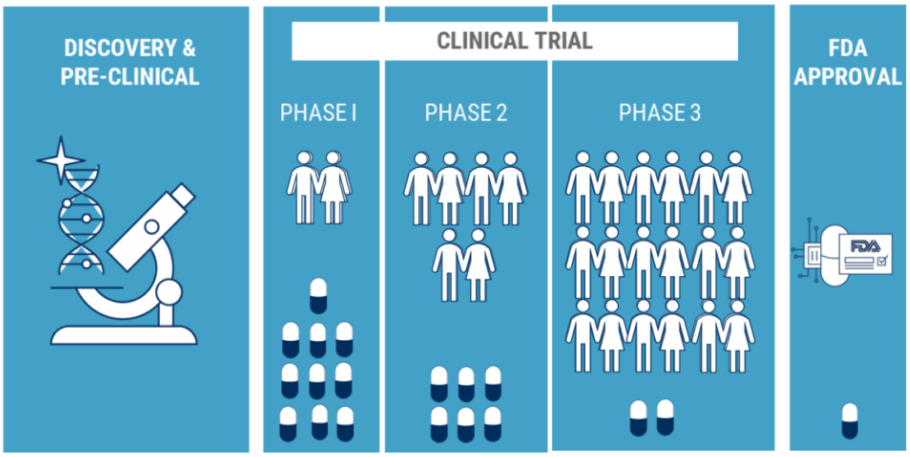
Reinforcement learning, a bit different

- 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

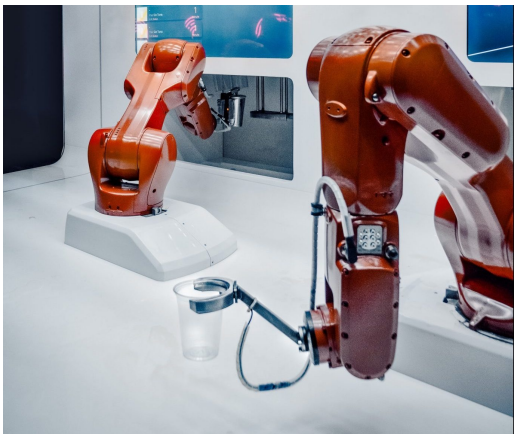
Reinforcement learning, some applications

- Design of clinical trials, Williamson (2019)



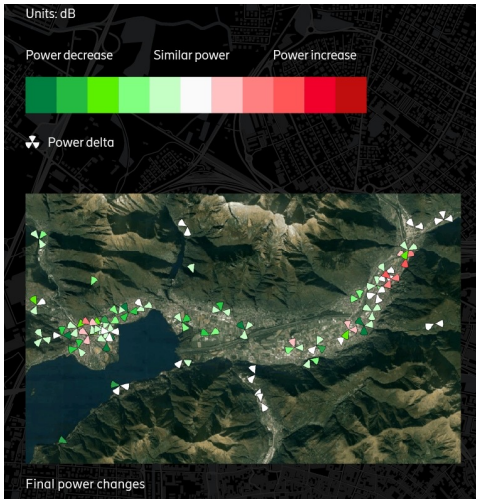
Reinforcement learning, some applications

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



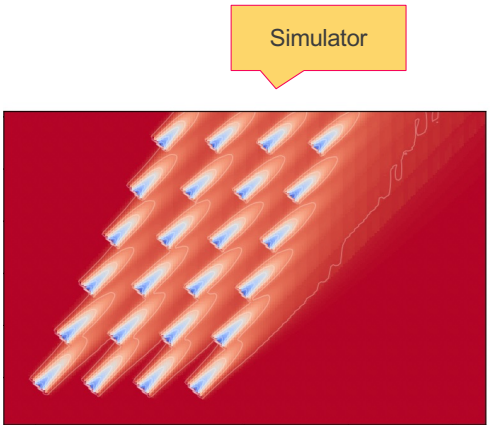
Reinforcement learning, some applications

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



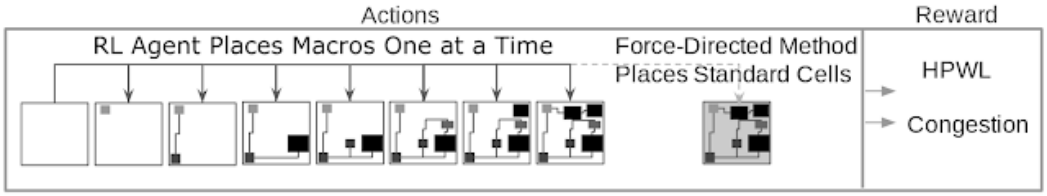
Reinforcement learning, some applications

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



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)

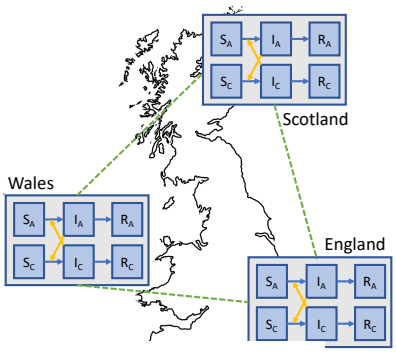


Reinforcement learning, some applications

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

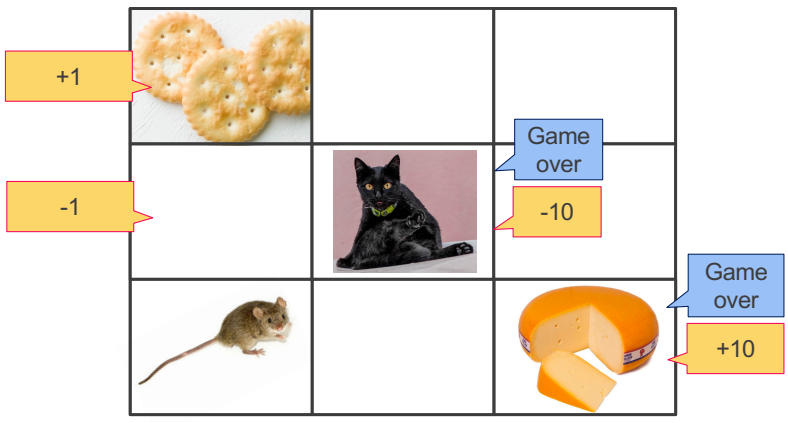
Reinforcement learning, some applications

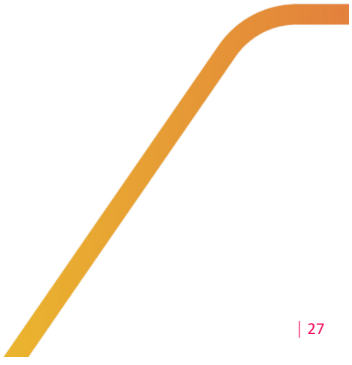
- Design of clinical trials, Williamson (2019)
- Robot control, Kober (2013)
- Telecom, *Ericsson* (2020)
- Wind farm control, Verstraeten (2020)
- Chip design, Mirhoseini (2020)
- Electric vehicle charging, *Energis/VUB* (2020)
- Epidemic control, Libin (2020)



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



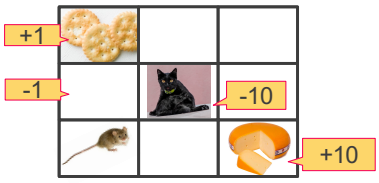


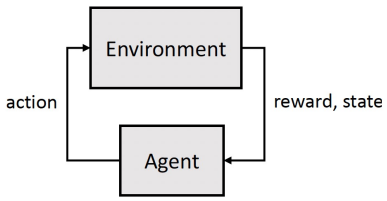
Markov decision process

Markovian

A Markov Decision Process corresponds to a tuple $\langle \mathcal{S}, \mathcal{A}, T, \gamma, R \rangle$, 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
- $T(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ 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
- $R(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ 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 $\langle \mathcal{S}, \mathcal{A}, T, \gamma, R \rangle$, 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
- $T(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ 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
- $R(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ 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 $\langle \mathcal{S}, \mathcal{A}, T, \gamma, R \rangle$, 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

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

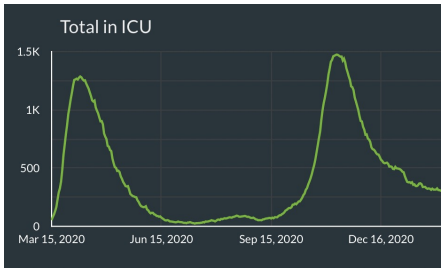
One trajectory through the MDP

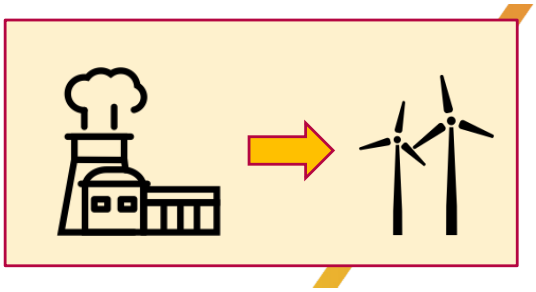
The return, or the discounted sum of rewards, starting from time step t, is defined as:

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

T is horizon

where γ is the discount factor.

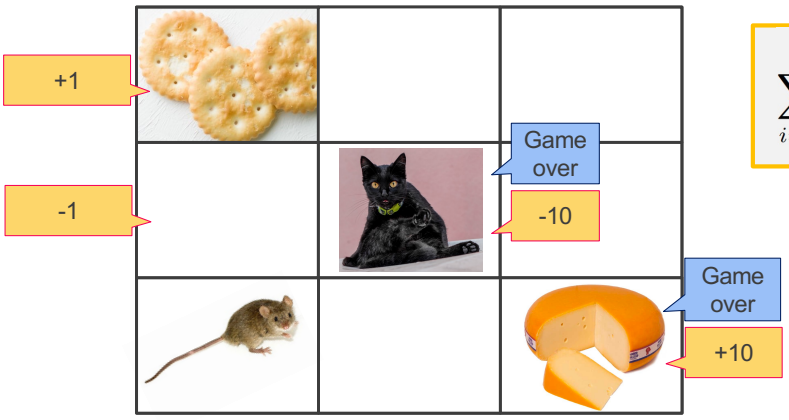




Policy

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

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

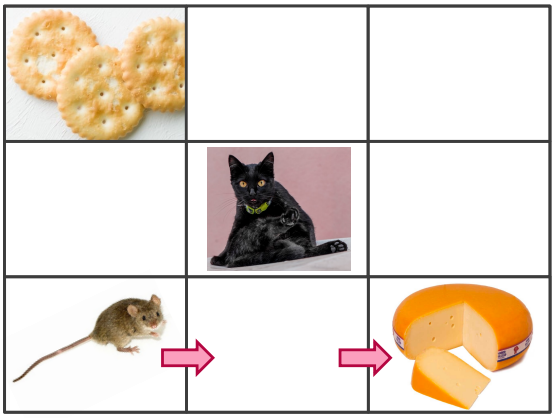


$$\sum_{i=0}^{T} \gamma^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right)$$

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 $\mathbf{s}$:

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

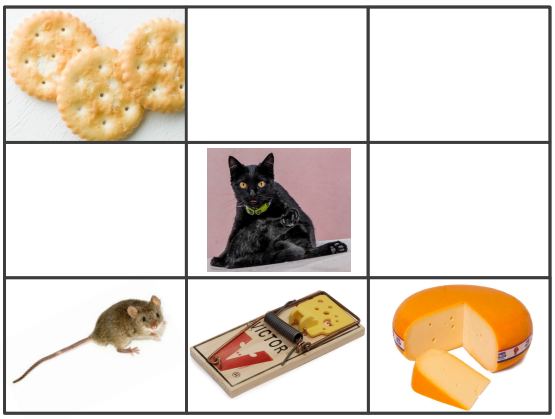


$p(.,F) = 1$

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 $\mathbf{s}$:

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



$p(.,.)$ = “Avoid cat”

Policies, some intuition

- What is a policy to you?
- How to treat a patient that is admitted to the hospital?

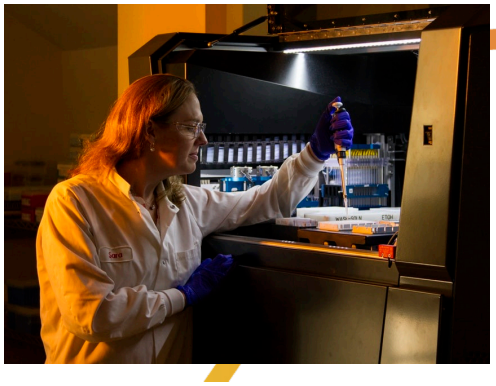
Policies, some intuition

- What is a policy to you?
- How to treat a patient that is admitted to the hospital?
- How to treat an HIV patient?

Policies, some intuition

- What is a policy to you?

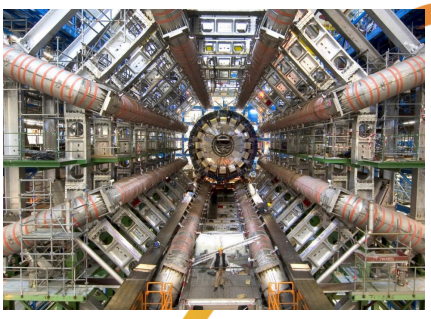
- How to treat a patient that is admitted to the hospital?
- How to treat an HIV patient?
- How to conduct a lab experiment?



Policies, some intuition

- What is a policy to you?

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



Value functions

- 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:

$$V^p(\mathbf{s}) = \mathop{\mathbb{E}}_{p,T}\left[\sum_{i=0}^{T}\gamma^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right) \mid \mathbf{s}_t = \mathbf{s}\right]$$

Value functions

- 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:

$$V^p(\mathbf{s}) = \mathop{\mathbb{E}}_{p,T}\left[\sum_{i=0}^{T}\gamma^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right) \mid \mathbf{s}_t = \mathbf{s}\right]$$

- 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:

$$Q^p(\mathbf{s}, \mathbf{a}) = \mathop{\mathbb{E}}_{p,T}\left[\sum_{i=0}^{T}\gamma^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right) \mid \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right]$$

Value function

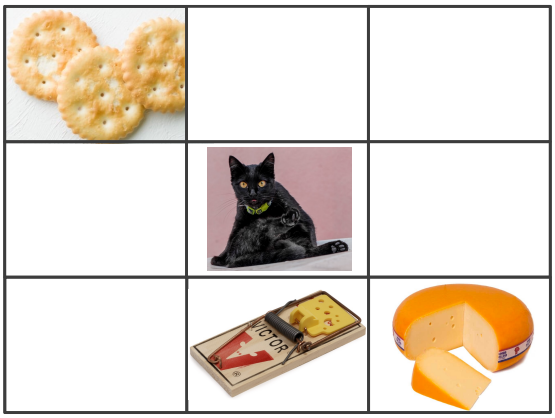
- Observation:
  - from an **optimal Q function**, we can derive an optimal policy

Q*(s,a)

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

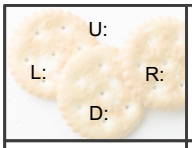
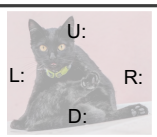
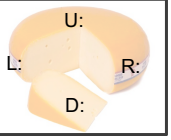
Q-learning

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



Q-learning

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

 U: L: R: D:	U: L: R: D:	U: L: R: D:
U: L: R: D:	 U: L: R: D:	U: L: R: D:
U: L: R: D:	U: L: R: D:	 U: L: R: D:

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?

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?

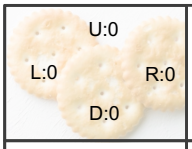
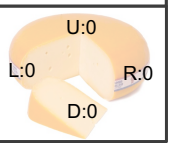
Exploitation vs Exploration

- 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

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

 U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0
U:0 L:0 R:0 D:0	 U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0
 U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0	 U:0 L:0 R:0 D:0

Q-learning

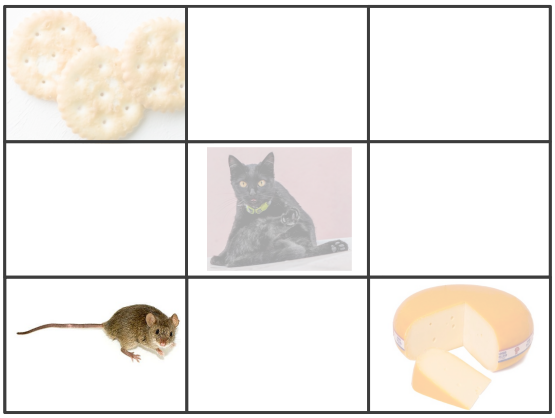
- Estimate Q*, by maintaining a Q-table:

Balance **exploration** and **exploitation**

U:0 L:0 R:0 D:-1	U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0
U:1 L:0 R:0 D:0	U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0
U:-1 L:0 R:0 D:0	U:0 L:0 R:0 D:0	U:0 L:0 R:0 D:0

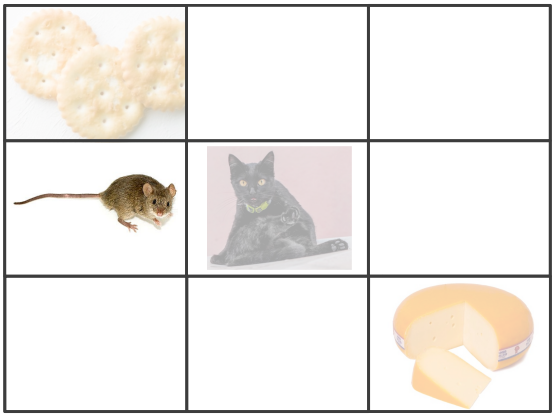
Q-learning

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



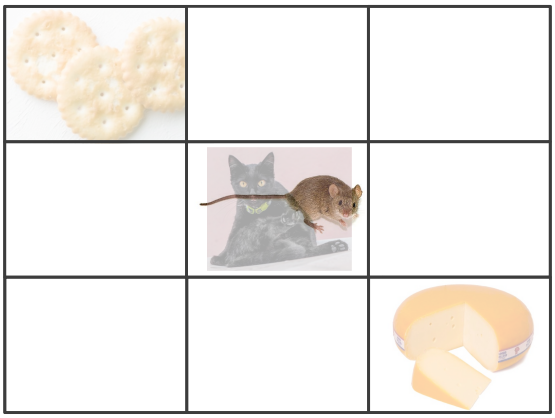
Q-learning

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



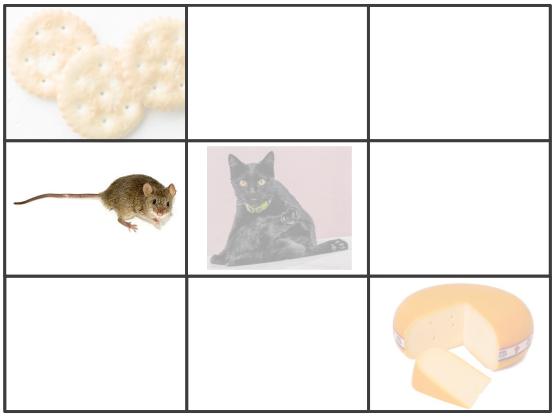
Q-learning

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



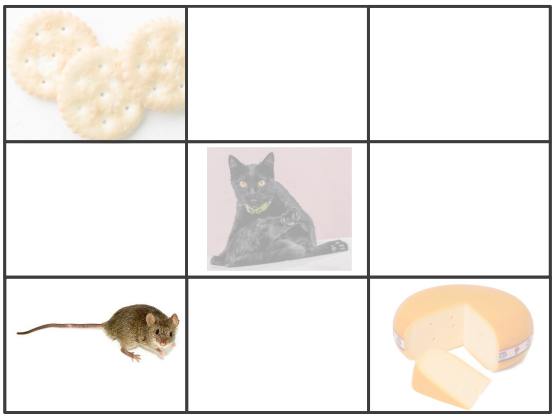
Q-learning

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



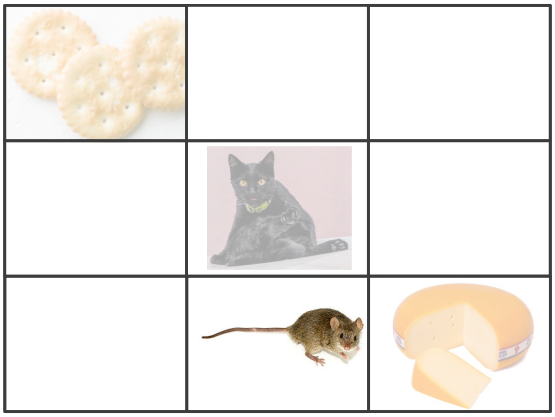
Q-learning

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



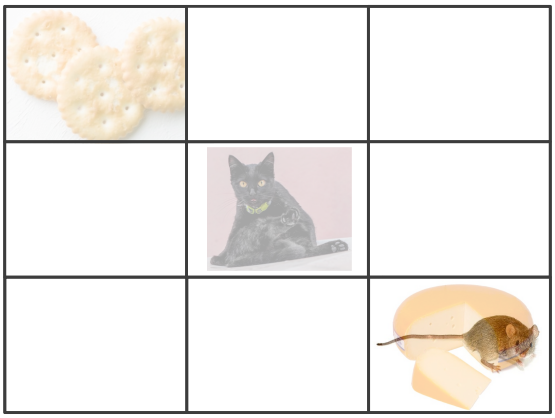
Q-learning

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



Q-learning

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



Q-learning

Watkins, 1999

Given: a learning rate α
$\forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$: initialize $\hat{Q}(\mathbf{s}, \mathbf{a})$
for *each episode* e **do**
  Initialize $\mathbf{s}_0$
  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

Q-learning

Given: a learning rate α
$\forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$: initialize $\hat{Q}(\mathbf{s}, \mathbf{a})$
for *each episode* e **do**
  Initialize $\mathbf{s}_0$
  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

Given: a learning rate α
$\forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$: initialize $\hat{Q}(\mathbf{s}, \mathbf{a})$
for *each episode* e **do**
  Initialize $\mathbf{s}_0$
  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

TD-learning

temporal difference

new value

old value

Q-learning

Convergence guarantees

TD-learning

Given: a learning rate α
$\forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$: initialize $\hat{Q}(\mathbf{s}, \mathbf{a})$
for *each episode* e **do**
  Initialize $\mathbf{s}_0$
  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

temporal difference

new value

old value

Q-learning

- Questions?
- One question for you:
  - “What is the disadvantage of using a Q-table?”

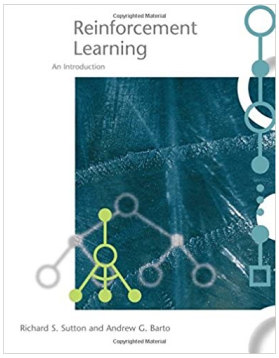
Reinforcement learning, some milestones

- Backgammon RL agent, Tesauro (1992)

Reinforcement learning, some milestones

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

RL bible

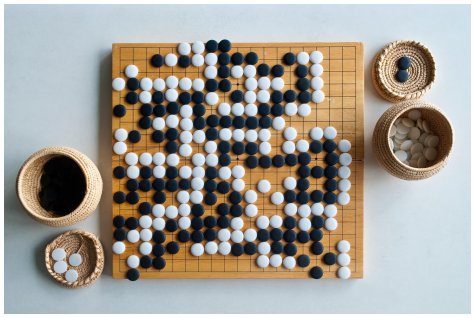


Reinforcement learning, some milestones

- 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)

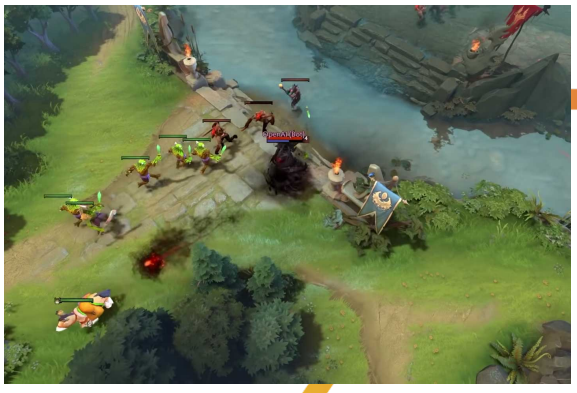


Reinforcement learning, some milestones

- Backgammon RL agent, Tesauro (1992)
- Reinforcement Learning book, Sutton and Barto (1998)
- Atari agent, *Deepmind* (2015)
- AlphaGo, *Deepmind* (2016)
- AlphaGo Zero, *Deepmind* (2017)

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)



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)

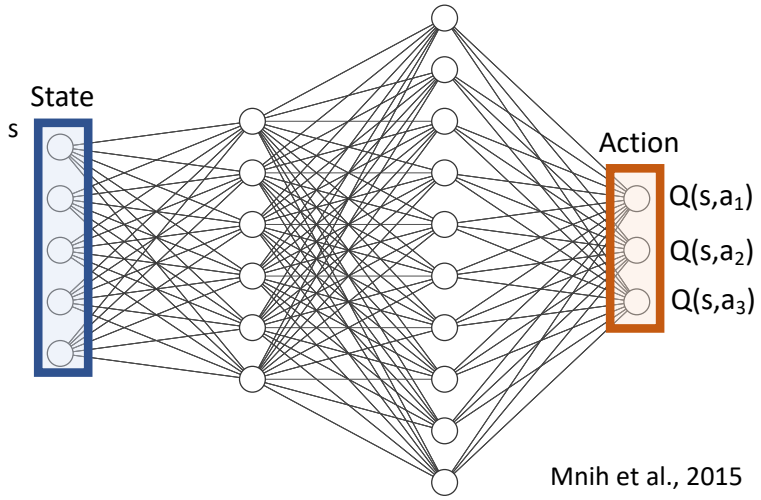
Q-learning in a large state space

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

Function approximation

Deep Q-networks (DQN)

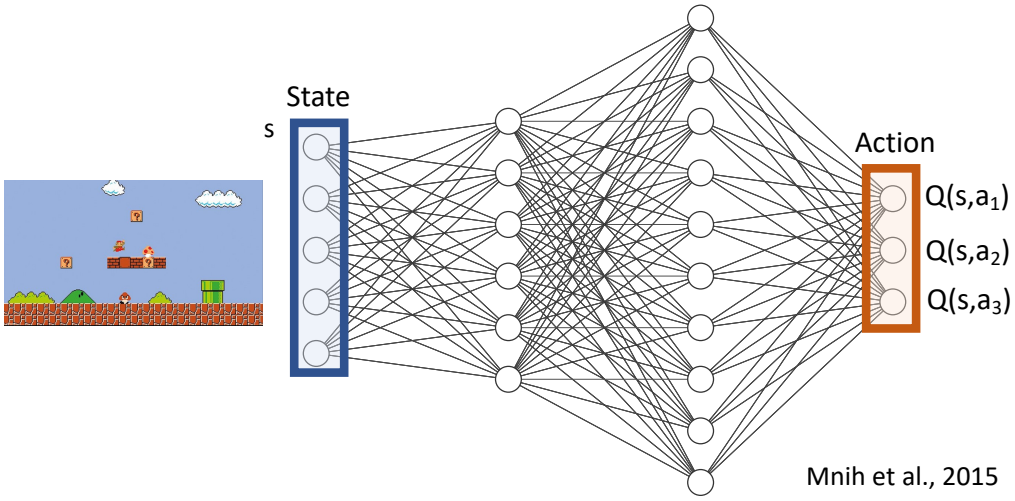
$$Q\left(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta}\right)$$



Mnih et al., 2015

Deep Q-networks (DQN)

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



Mnih et al., 2015

Deep Q-networks

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

- Minimize temporal difference error, from mini-batch:

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

- We could use the mean squared error:

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

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

Unstable

Deep Q-networks

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

- Minimize temporal difference error, from mini-batch:

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

- We could use the mean squared error:

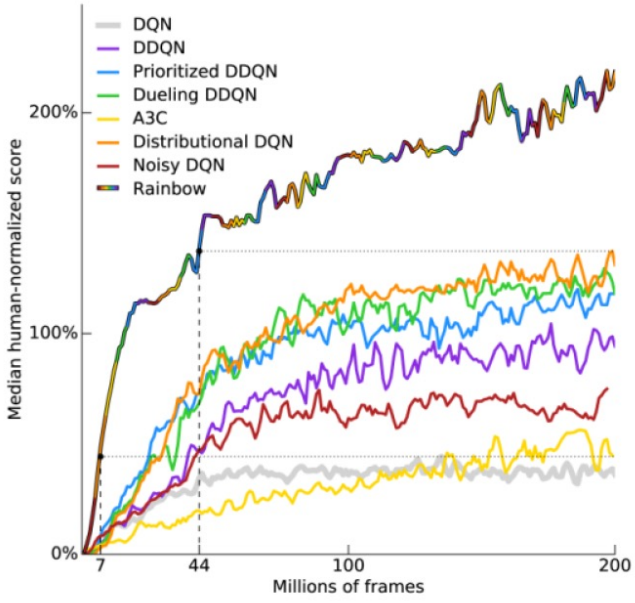
$$L\left(\mathbf{s}, \mathbf{a}\right) = \frac{1}{N} \sum_i \left(y_i - Q\left(\mathbf{s}_i, \mathbf{a}_i \mid \boldsymbol{\theta}\right)\right)^2 \text{ , where,}$$

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

Use a separate target network

Unstable

Deep Q-networks



Hessel, 2017

Deep Q-networks

- Questions?
- What about large/continuous action spaces? What kind of problem do you see?

Policy gradient

- Parameterize a policy directly $p_{\boldsymbol{\theta}}$
- Optimize the expectation of the policy gradient:

$$\mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}},T}\left[\sum_{t=0}^{\infty}\overbrace{\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}^{\text{log prob. of the policy's network output}}\underbrace{\hat{A}_t}_{\text{Advantage}}\right]$$

Gradient of the policy

Policy gradient

- Parameterize a policy directly $p_{\boldsymbol{\theta}}$
- Optimize the expectation of the policy gradient:

$$\mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}},T}\left[\sum_{t=0}^{\infty}\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\hat{A}_t\right]$$

$$\hat{A}_t = \underbrace{\left[\sum_{i=0}^{\infty} d^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right)\right]}_{\text{Return}} - \underbrace{V^p\left(\mathbf{s}_t\right)}_{\text{Value function}}$$

Policy gradient

- Parameterize a policy directly $p_{\boldsymbol{\theta}}$
- Optimize the expectation of the policy gradient:

$$\mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}},T}\left[\sum_{t=0}^{\infty}\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\hat{A}_t\right]$$

$$\hat{A}_t = \left[\sum_{i=0}^{\infty} d^i R\left(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}, \mathbf{s}_{t+i+1}\right)\right] - V^p\left(\mathbf{s}_t\right)$$

- Estimate using:

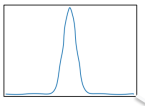
$$\hat{g} = \frac{1}{|Y|}\sum_{y\in Y}\sum_{t}\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\hat{A}_t$$

Policy gradient

Williams, 1992

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
  Re-fit $V^p(s_t)$
  Compute the policy gradient estimate $\hat{g}_i$
  Update the policy
end

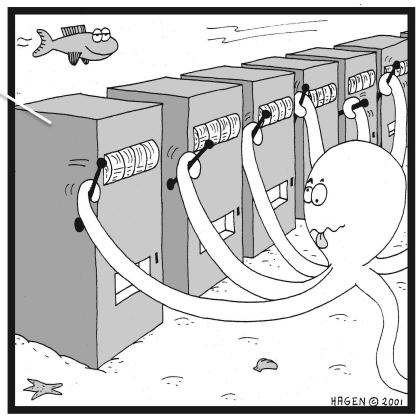
Multi-armed bandits

$\mathcal{R}_k$ 

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

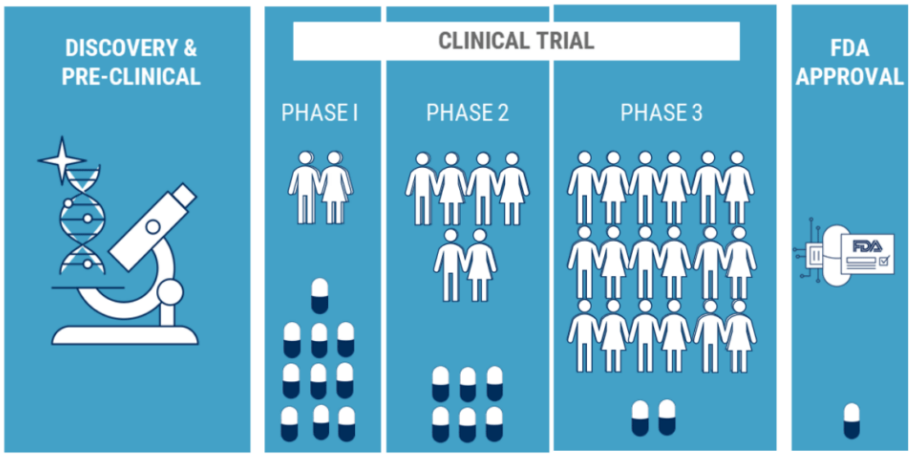
Stochastic reward: r_k

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



Multi-armed bandit application

- Clinical trials



Cumulative regret

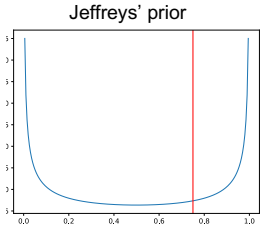
- ϵ - greedy
  - Exploit with probability $1 - \epsilon$
- UCB
  - $A_t = \text{argmax}_k \left[\hat{\mu}_k + c \sqrt{\frac{\ln(t)}{N_t(k)}} \right]$

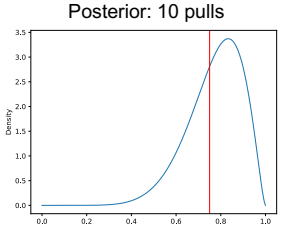
Use prior knowledge: Bayesian philosophy

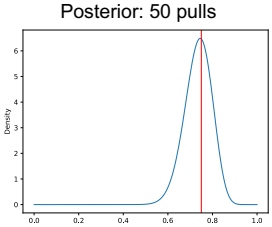
- It's all about belief
  - Prior belief
  - Updating our belief, given data

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

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

(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**

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

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

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

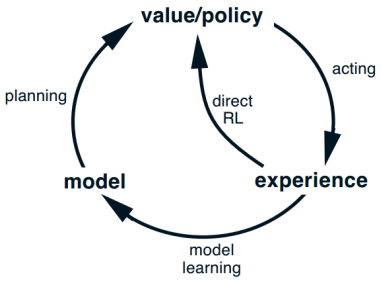
$\mathcal{H}^{(t)} \leftarrow \mathcal{H}^{(t-1)} \cup \{a^{(t)}, r^{(t)}\}$

end

(Russo, 2016)

Things we did *not* talk about

- Mixing planning and learning



Mosic, 2017

Things we did *not* talk about

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

Reading material

- RL book by Sutton and Barto
  - Mixing planning and learning
- Connecting RL with biomed:
  - Guiding the mitigation of epidemics with RL, Libin, 2020
  - https://ai.vub.ac.be/education/research/phd-theses
- 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

Thank you! Any more questions?

ai.vub.ac.be

@aibrussels

Pieter.Libin@vub.be

@PieterLibin