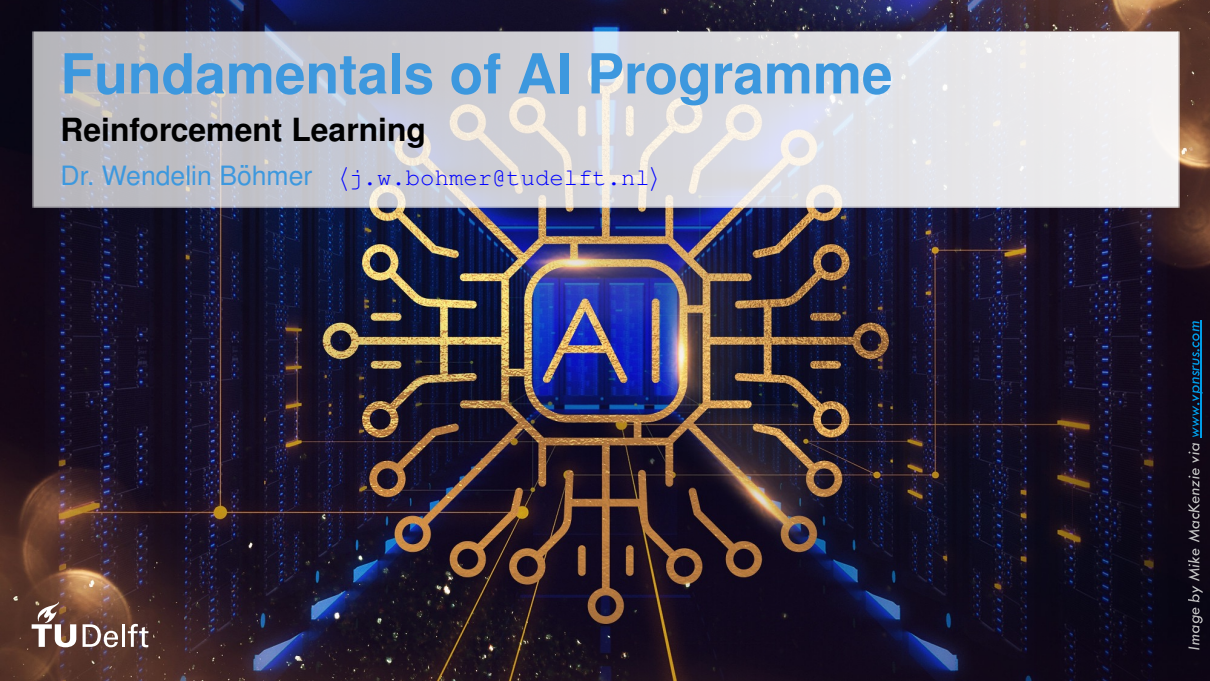
Fundamentals of AI Programme

Reinforcement Learning

Dr. Wendelin Böhmer ⟨j.w.bohmer@tudelft.nl⟩



TU Delft

Image by Mike MacKenzie via www.vpnsrus.com

Content of this lecture

1. Learning to Interact
2. Tabular Q-learning
3. Deep Reinforcement Learning
4. State-of-the-Art and Limits

1

Reinforcement Learning: Learning to Interact

AI methods for interaction

- Search & optimization
  - task: reach goal with minimal costs
  - given: dynamics model of environment
  - prediction: action sequence that reaches goal

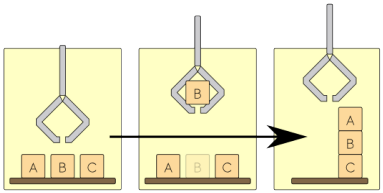
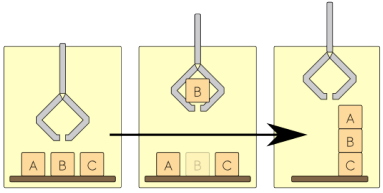


image source: www.wikipedia.org

AI methods for interaction

- Search & optimization
  - task: reach goal with minimal costs
  - given: dynamics model of environment
  - prediction: action sequence that reaches goal



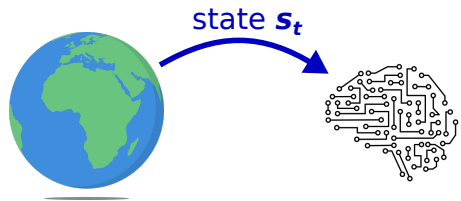
- But what if we do **not know** the environment?
  - go boldly where no one has gone before
  - uncertain dynamics or observations
  - unknown dynamics, e.g. humans in the loop

- Sometimes we need to **learn** from interaction!

image source: Steven Shorrock

Markov decision processes

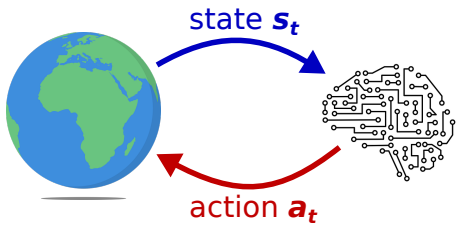
- MDP $M = \langle \mathcal{S}, \mathcal{A}, \rho, P, r \rangle$ consists of:
  - state space $\mathcal{S}$ of environment configurations
  - distribution $\rho(s_0)$ of initial states $s_0 \in \mathcal{S}$



see Chapter 3 in Sutton and Barto (2018) for a more formal introduction

Markov decision processes

- MDP $M = \langle \mathcal{S}, \mathcal{A}, \rho, P, r \rangle$ consists of:
  - state space $\mathcal{S}$ of environment configurations
  - distribution $\rho(s_0)$ of initial states $s_0 \in \mathcal{S}$
  - action space $\mathcal{A}$ of possible agent choices



see Chapter 3 in Sutton and Barto (2018) for a more formal introduction

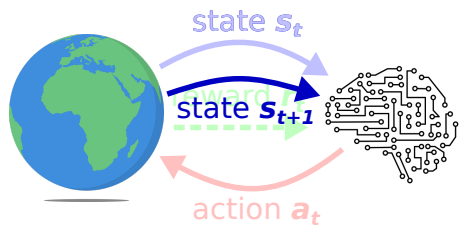
Markov decision processes

- MDP $M = \langle \mathcal{S}, \mathcal{A}, \rho, P, r \rangle$ consists of:
  - state space $\mathcal{S}$ of environment configurations
  - distribution $\rho(s_0)$ of initial states $s_0 \in \mathcal{S}$
  - action space $\mathcal{A}$ of possible agent choices
  - reward function $r(s_t, a_t)$ for choosing good or bad actions $a_t \in \mathcal{A}$ in $s_t \in \mathcal{S}$

see Chapter 3 in Sutton and Barto (2018) for a more formal introduction

Markov decision processes

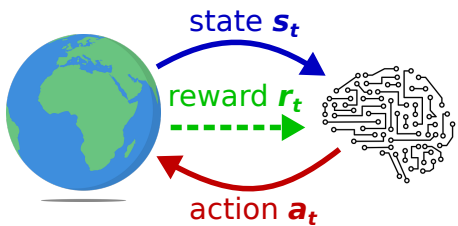
- MDP $M = \langle \mathcal{S}, \mathcal{A}, \rho, P, r \rangle$ consists of:
  - state space $\mathcal{S}$ of environment configurations
  - distribution $\rho(s_0)$ of initial states $s_0 \in \mathcal{S}$
  - action space $\mathcal{A}$ of possible agent choices
  - reward function $r(s_t, a_t)$ for choosing good or bad actions $a_t \in \mathcal{A}$ in $s_t \in \mathcal{S}$
- distribution $P(s_{t+1}|s_t, a_t)$ of transitions from $s_t \in \mathcal{S}$ to $s_{t+1} \in \mathcal{S}$ after choosing $a_t \in \mathcal{A}$



see Chapter 3 in Sutton and Barto (2018) for a more formal introduction

Markov decision processes

- MDP $M = \langle \mathcal{S}, \mathcal{A}, \rho, P, r \rangle$ consists of:
  - state space $\mathcal{S}$ of environment configurations
  - distribution $\rho(s_0)$ of initial states $s_0 \in \mathcal{S}$
  - action space $\mathcal{A}$ of possible agent choices
  - reward function $r(s_t, a_t)$ for choosing good or bad actions $a_t \in \mathcal{A}$ in $s_t \in \mathcal{S}$
  - distribution $P(s_{t+1}|s_t, a_t)$ of transitions from $s_t \in \mathcal{S}$ to $s_{t+1} \in \mathcal{S}$ after choosing $a_t \in \mathcal{A}$

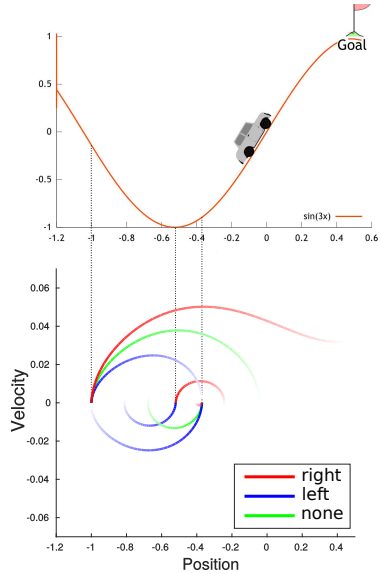


- Reinforcement learning aims for **actions** that maximize **all future reward**
  - future reward depends where we end up in the future → complex problem
  - no prior knowledge of reward or transition model

see Chapter 3 in Sutton and Barto (2018) for a more formal introduction

Example: MountainCar

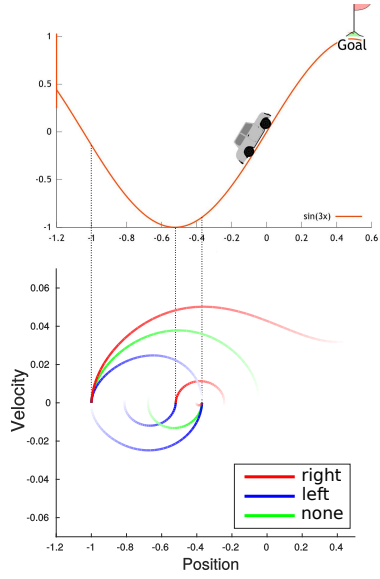
- Underactuated car in a valley between mountains
  - state is position and velocity: $\boldsymbol{s}_t = [x_t, v_t] \in \mathcal{S} \subset \mathbb{R}^2$
  - actions accelerate car $\boldsymbol{a}_t \in \mathcal{A} = \{\text{left}, \text{none}, \text{right}\}$
  - physics dynamics P: gravitation, but no friction



a reference implementation can be found at www.gymlibrary.ml

Example: MountainCar

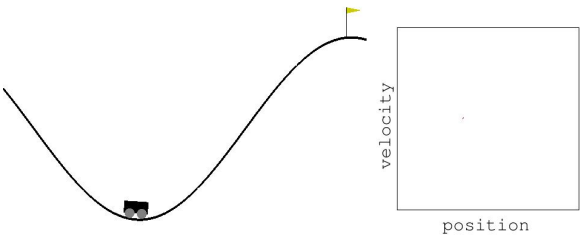
- Underactuated car in a valley between mountains
  - state is position and velocity: $\boldsymbol{s}_t = [x_t, v_t] \in \mathcal{S} \subset \mathbb{R}^2$
  - actions accelerate car $\boldsymbol{a}_t \in \mathcal{A} = \{\text{left}, \text{none}, \text{right}\}$
  - physics dynamics P: gravitation, but no friction
  - reward of 1 for reaching the flag, otherwise 0

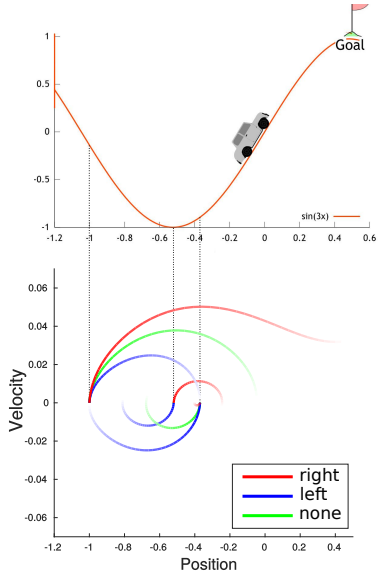


a reference implementation can be found at www.gymlibrary.ml

Example: MountainCar

- Underactuated car in a valley between mountains
  - state is position and velocity: $\boldsymbol{s}_t = [x_t, v_t] \in \mathcal{S} \subset \mathbb{R}^2$
  - actions accelerate car $\boldsymbol{a}_t \in \mathcal{A} = \{\text{left}, \text{none}, \text{right}\}$
  - physics dynamics P: gravitation, but no friction
  - reward of 1 for reaching the flag, otherwise 0
  - underactuated car needs to pick up speed first

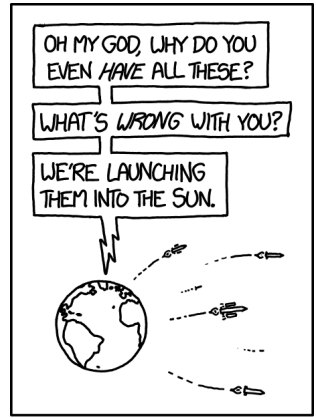




a reference implementation can be found at www.gymlibrary.ml

Learning to interact

- In this video we have:
  - discussed under which situations we need to learn interaction
  - learned how a stochastic environment is formalized as a MDP
  - familiarized ourselves with the MountainCar environment



THE MOMENT THE COMPUTERS CONTROLLING OUR NUCLEAR ARSENALS BECAME SENTIENT

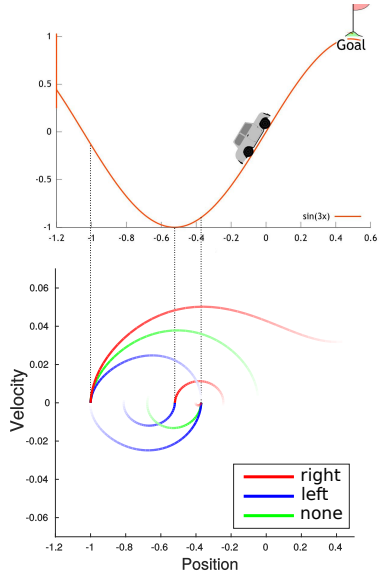
image source: xkcd.com

2

Reinforcement Learning: Tabular Q-learning

Agent policies

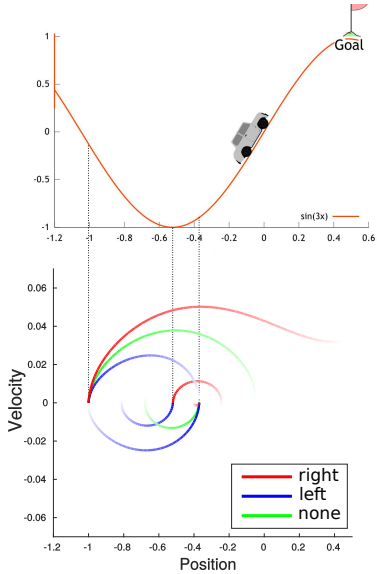
- Policy π defines agent's behavior
  - $\pi(a|s)$ probability the agent chooses $a \in \mathcal{A}$ in $s \in \mathcal{S}$



Agent policies

- Policy π defines agent's behavior
  - $\pi(a|s)$ probability the agent chooses $a \in \mathcal{A}$ in $s \in \mathcal{S}$
  - allows expectations over all possible futures

$$\mathbb{E}_\pi\Big[\dots\Big] \quad := \quad \mathbb{E}\Big[\dots \Big| \begin{matrix} s_0 \sim \rho(\cdot), \quad r_t = r(s_t, a_t) \\ a_t \sim \pi(\cdot|s_t),\ s_{t+1} \sim P(\cdot|s_t, a_t) \end{matrix}, \ \forall t \geq 0\Big]$$

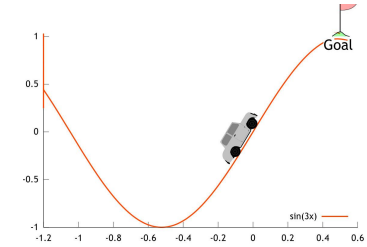


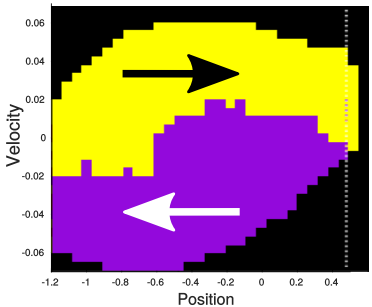
Agent policies

- Policy π defines agent's behavior
  - $\pi(a|s)$ probability the agent chooses $a \in \mathcal{A}$ in $s \in \mathcal{S}$
  - allows expectations over all possible futures

$$\mathbb{E}_\pi\Big[\dots\Big] \quad := \quad \mathbb{E}\Big[\dots \Big| \begin{matrix} s_0 \sim \rho(\cdot), & r_t = r(s_t, a_t) \\ a_t \sim \pi(\cdot|s_t), & s_{t+1} \sim P(\cdot|s_t, a_t) \end{matrix}, \ \forall t \geq 0\Big]$$

- Discretize continuous states $s \in \mathcal{S}$
  - policy table/matrix $\Pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ with $\Pi_{sa} := \pi(a|s)$





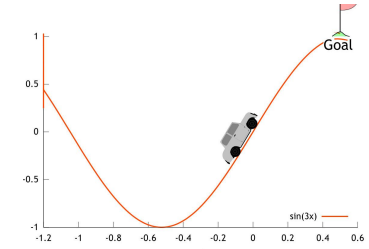
Agent policies

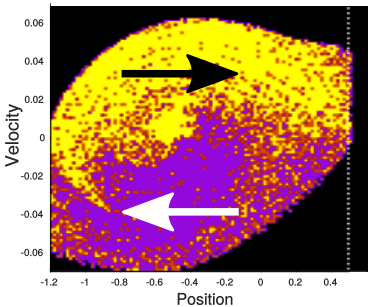
- Policy π defines agent's behavior
  - $\pi(a|s)$ probability the agent chooses $a \in \mathcal{A}$ in $s \in \mathcal{S}$
  - allows expectations over all possible futures

$$\mathbb{E}_\pi\Big[\dots\Big] \quad := \quad \mathbb{E}\Big[\dots \Big| \begin{matrix} s_0 \sim \rho(\cdot), & r_t = r(s_t, a_t) \\ a_t \sim \pi(\cdot|s_t),\ s_{t+1} \sim P(\cdot|s_t, a_t) \end{matrix}, \ \forall t \geq 0 \Big]$$

- Discretize continuous states $s \in \mathcal{S}$
  - policy table/matrix $\Pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ with $\Pi_{sa} := \pi(a|s)$

- resolution determines control precision

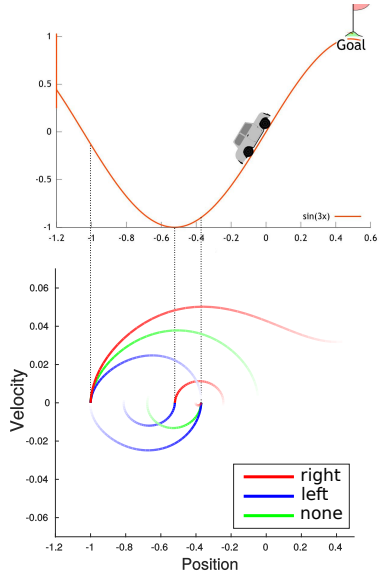




Policy evaluation

- Policy should maximize the **value** V^π
  - expected sum of discounted future rewards

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0=s\right] \qquad ,\gamma \in [0,1)$$

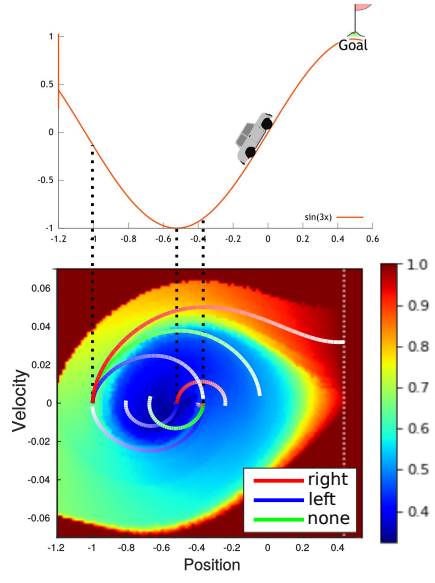


the infinite sum is a geometric series with $\sum_{t=0}^{\infty}\gamma^t = \frac{1}{1-\gamma}$

Policy evaluation

- Policy should maximize the **value** V^{π}
  - expected sum of discounted future rewards
  - same discretization $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|}$ as policy

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r_t \middle| s_0 = s\right] \qquad , \gamma \in [0, 1)$$

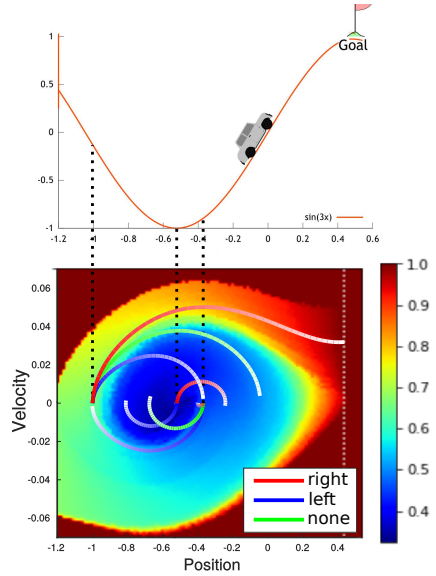


Policy evaluation

- Policy should maximize the **value** V^π
  - expected sum of discounted future rewards
  - same discretization $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|}$ as policy

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t \,\middle|\, s_0{=}s\right] \qquad ,\gamma \in [0,1)$$

$$= \mathbb{E}_\pi\left[r_0 + \underbrace{\mathbb{E}_\pi[\gamma r_1 + \gamma^2 r_2 + \ldots \,|s_1]}_{\gamma V^\pi(s_1)} \,\middle|\, s_0{=}s\right]$$



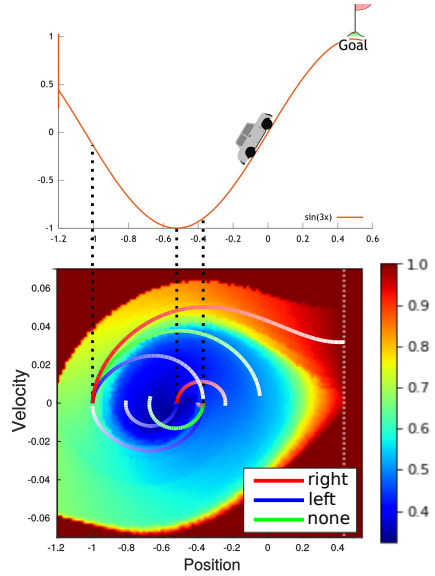
this equation goes back to Richard Bellman (1957), see Section 3.5 in Sutton and Barto (2018)

Policy evaluation

- Policy should maximize the **value** V^π
  - expected sum of discounted future rewards
  - same discretization $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|}$ as policy

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t \,\middle|\, s_0{=}s\right] && ,\gamma \in [0,1) \\
&= \mathbb{E}_\pi\Big[r_0 + \underbrace{\mathbb{E}_\pi[\gamma r_1 + \gamma^2 r_2 + \ldots \,|s_1]}_{\gamma V^\pi(s_1)} \,\Big|\, s_0{=}s\Big] \\
&= \mathbb{E}_\pi\left[r_0 + \gamma V^\pi(s_1) \,\middle|\, s_0{=}s\right]
\end{aligned}
$$

- Recursive **Bellman equation** of the state value



this equation goes back to Richard Bellman (1957), see Section 3.5 in Sutton and Barto (2018)

The optimal Bellman equation

$$\max_{\pi} V^{\pi}(s) \quad = \quad \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s\right]$$

The optimal Bellman equation

$$\max_{\pi} V^{\pi}(s) \quad = \quad \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s\right] \quad = \quad \overbrace{\max_{a} \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, \begin{matrix} s_0 = s \\ a_0 = a \end{matrix}\right]}^{Q^*(s,a)}$$

The optimal Bellman equation

$$\max_{\pi} V^{\pi}(s) = \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0{=}s\right] = \overbrace{\max_{a}\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, \begin{matrix} s_0{=}s \\ a_0{=}a \end{matrix}\right]}^{Q^*(s,a)}$$

$$\overset{(*)}{=} \max_{a} \mathbb{E}\left[r(s_0,a_0) + \gamma \max_{\pi} V^{\pi}(s_1)\,\middle|\, \begin{matrix} s_0{=}s \\ a_0{=}a \end{matrix}\right]$$

$^{(*)}$ $\max_\pi \mathbb{E}[\ldots] = \mathbb{E}[\max_\pi \ldots]$ only holds because a policy π^* exists that is optimal in all states: $V^{\pi^*}(s) \geq V^{\pi}(s),\ \forall \pi, \forall s \in \mathcal{S}$.

The optimal Bellman equation

$$
\begin{aligned}
\max_{\pi} V^{\pi}(s) &= \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0{=}s\right] = \overbrace{\max_{a}\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, \begin{smallmatrix} s_0{=}s \\ a_0{=}a \end{smallmatrix}\right]}^{Q^*(s,a)} \\
&\overset{(*)}{=} \max_{a} \mathbb{E}\left[r(s_0,a_0) + \gamma \max_{\pi} V^{\pi}(s_1)\,\middle|\, \begin{smallmatrix} s_0{=}s \\ a_0{=}a \end{smallmatrix}\right] \\
&= \max_{a} \mathbb{E}\left[r(s_0,a_0) + \gamma \underbrace{\max_{a'}\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s'_t,a'_t)\,\middle|\, \begin{smallmatrix} s'_0{=}s_1 \\ a'_0{=}a' \end{smallmatrix}\right]}_{Q^*(s_1,a')}\,\middle|\, \begin{smallmatrix} s_0{=}s \\ a_0{=}a \end{smallmatrix}\right]
\end{aligned}
$$

$^{(*)}$ $\max_{\pi}\mathbb{E}[\ldots] = \mathbb{E}[\max_{\pi}\ldots]$ only holds because a policy π^* exists that is optimal in all states: $V^{\pi^*}(s) \geq V^{\pi}(s),\ \forall\pi, \forall s \in \mathcal{S}$.

The optimal Bellman equation

$$
\begin{aligned}
\max_\pi V^\pi(s) &= \max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0{=}s\right] = \overbrace{\max_a \max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, \begin{smallmatrix}s_0{=}s\\ a_0{=}a\end{smallmatrix}\right]}^{Q^*(s,a)} \\
&\overset{(*)}{=} \max_a \mathbb{E}\left[r(s_0,a_0) + \gamma \max_\pi V^\pi(s_1)\,\middle|\, \begin{smallmatrix}s_0{=}s\\ a_0{=}a\end{smallmatrix}\right] \\
&= \max_a \mathbb{E}\left[r(s_0,a_0) + \gamma \max_{a'} \underbrace{\max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s'_t,a'_t)\,\middle|\, \begin{smallmatrix}s'_0{=}s_1\\ a'_0{=}a'\end{smallmatrix}\right]}_{Q^*(s_1,a')}\,\middle|\, \begin{smallmatrix}s_0{=}s\\ a_0{=}a\end{smallmatrix}\right]
\end{aligned}
$$

- Optimal state-action **Q-values**:

$$
Q^*(s,a) = r(s,a) + \gamma\,\mathbb{E}\left[\max_{a'} Q^*(s',a')\,\middle|\, s'\sim P(\cdot|s,a)\right]
$$

see Section 3.6 in Sutton and Barto (2018)

The optimal Bellman equation

$$
\begin{aligned}
\max_{\pi} V^{\pi}(s) &= \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0{=}s\right] = \overbrace{\max_{a}\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, \begin{smallmatrix} s_0{=}s \\ a_0{=}a \end{smallmatrix}\right]}^{Q^*(s,a)} \\
&\overset{(*)}{=} \max_{a} \mathbb{E}\left[r(s_0,a_0) + \gamma \max_{\pi} V^{\pi}(s_1)\,\middle|\, \begin{smallmatrix} s_0{=}s \\ a_0{=}a \end{smallmatrix}\right] \\
&= \max_{a} \mathbb{E}\left[r(s_0,a_0) + \gamma \max_{a'} \underbrace{\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s'_t,a'_t)\,\middle|\, \begin{smallmatrix} s'_0{=}s_1 \\ a'_0{=}a' \end{smallmatrix}\right]}_{Q^*(s_1,a')}\,\middle|\, \begin{smallmatrix} s_0{=}s \\ a_0{=}a \end{smallmatrix}\right]
\end{aligned}
$$

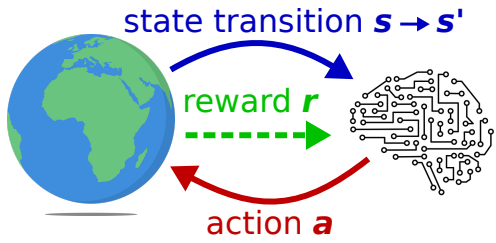
- Optimal state-action **Q-values** and **policy**:

$$
\begin{aligned}
Q^*(s,a) &= r(s,a) + \gamma\, \mathbb{E}\left[\max_{a'} Q^*(s',a')\,\middle|\, s'\sim P(\cdot|s,a)\right] \\
\pi^*(a|s) &= \begin{cases} 1, & \text{if } \; a = \arg\max_{a'} Q^*(s,a') \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
$$

see Section 3.6 in Sutton and Barto (2018)

Tabular Q-learning

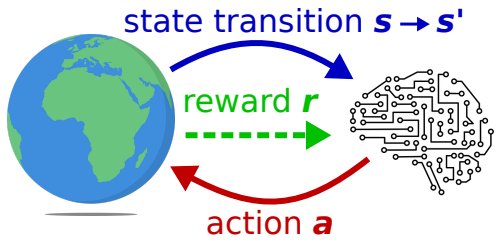
- Estimate optimal Q-value Q^*
  - while interacting with environment
  - MDP is unknown, can only be sampled



tabular Q-learning has been introduced by Watkins and Dayan (1992)

Tabular Q-learning

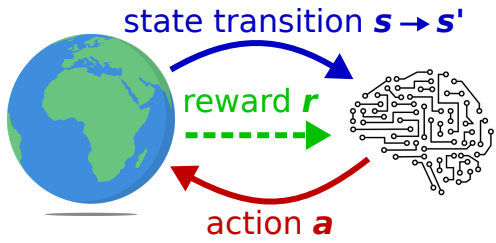
- Estimate optimal Q-value Q^*
  - while interacting with environment
  - MDP is unknown, can only be sampled
- Table $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ of finite $\mathcal{S} \times \mathcal{A}$
  - same discretization as policy



tabular Q-learning has been introduced by Watkins and Dayan (1992)

Tabular Q-learning

- Estimate optimal Q-value Q^*
  - while interacting with environment
  - MDP is unknown, can only be sampled
- Table $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ of finite $\mathcal{S} \times \mathcal{A}$
  - same discretization as policy



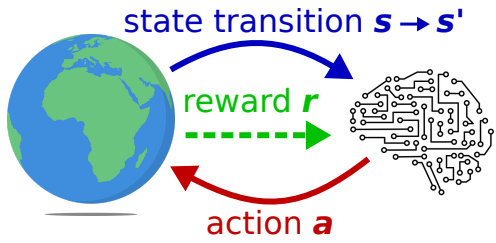
- Update table $\mathbf{Q}$ after observing transition $s, a \to r, s'$
  - temporal averaging with learning rate $\alpha \in (0, 1]$

$$Q_{sa} \leftarrow (1-\alpha)\underbrace{Q_{sa}}_{\text{old value}} + \alpha\underbrace{\left(r + \gamma \max_{a'} Q_{s'a'}\right)}_{\text{Bellman update}}$$

tabular Q-learning has been introduced by Watkins and Dayan (1992)

Tabular Q-learning

- Estimate optimal Q-value Q^*
  - while interacting with environment
  - MDP is unknown, can only be sampled
- Table $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ of finite $\mathcal{S} \times \mathcal{A}$
  - same discretization as policy



- Update table $\mathbf{Q}$ after observing transition $s, a \to r, s'$
  - temporal averaging with learning rate $\alpha \in (0, 1]$

$$Q_{sa} \leftarrow (1-\alpha)\underbrace{Q_{sa}}_{\text{old value}} + \alpha\underbrace{\left(r + \gamma \max_{a'} Q_{s'a'}\right)}_{\text{Bellman update}} = Q_{sa} + \alpha\underbrace{\left(r + \gamma \max_{a'} Q_{s'a'} - Q_{sa}\right)}_{\text{temporal difference (TD) error}}$$

tabular Q-learning has been introduced by Watkins and Dayan (1992)

Tabular Q-learning

- Estimate optimal Q-value Q^*
  - while interacting with environment
  - MDP is unknown, can only be sampled
- Table $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ of finite $\mathcal{S} \times \mathcal{A}$
  - same discretization as policy

```
1 s' = environment.reset()
2 while not done:
3     s, a = s', argmax(Q[s'])
4     s', r, done = environment.step(a)
5     td = r + γ * max(Q[s']) - Q[s, a]
6     Q[s, a] += α * td
```

- Update table $\mathbf{Q}$ after observing transition $s, a \to r, s'$
  - temporal averaging with learning rate $\alpha \in (0, 1]$

$$Q_{sa} \leftarrow (1-\alpha) \underbrace{Q_{sa}}_{\text{old value}} + \alpha \underbrace{\left(r + \gamma \max_{a'} Q_{s'a'} \right)}_{\text{Bellman update}} = Q_{sa} + \alpha \underbrace{\left(r + \gamma \max_{a'} Q_{s'a'} - Q_{sa} \right)}_{\text{temporal difference (TD) error}}$$

tabular Q-learning has been introduced by Watkins and Dayan (1992)

Exploration and exploitation

- Tabular Q-learning converges to optimal Q^*
  - if all state-action pairs are visited often enough
  - how can we guarantee that we **explore** enough?

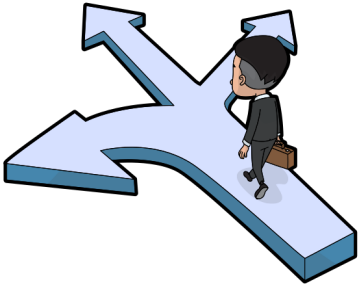


image source: wikipedia.org

Exploration and exploitation

- Tabular Q-learning converges to optimal Q^*
  - if all state-action pairs are visited often enough
  - how can we guarantee that we **explore** enough?

1. ϵ-greedy: explore randomly with probability ϵ

$$\pi_\epsilon(a|s) \quad := \quad (1-\epsilon)\underbrace{\pi^*(a|s)}_{\text{greedy}} + \epsilon\underbrace{\frac{1}{|\mathcal{A}|}}_{\text{random}}$$

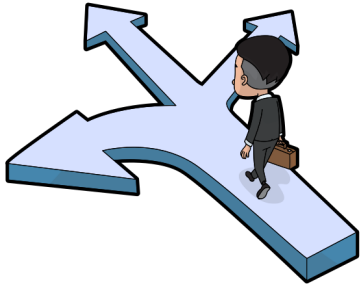


image source: wikipedia.org

Exploration and exploitation

- Tabular Q-learning converges to optimal Q^*
  - if all state-action pairs are visited often enough
  - how can we guarantee that we **explore** enough?

1. ϵ-greedy: explore randomly with probability ϵ

$$\pi_\epsilon(a|s) \quad := \quad (1-\epsilon)\underbrace{\pi^*(a|s)}_{\text{greedy}} + \epsilon\underbrace{\frac{1}{|\mathcal{A}|}}_{\text{random}}$$

2. optimistic initialization: overestimate values
   - initialize $Q_{sa} > \max_{s',a'} Q^*(s',a')$
   - updates reduce table entries towards $Q^*(s,a)$
   - under-explored actions are more attractive

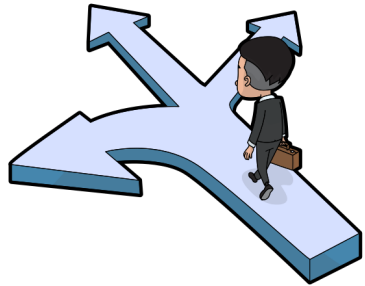


image source: wikipedia.org

Tabular Q-learning

- In this video we have:
  - defined the agent's behavior as policy π
  - defined how to evaluate a policy V^{π}
  - derived the value Q^{*} of the optimal policy
  - introduced the tabular Q-learning algorithm
  - discussed how to guarantee sufficient exploration

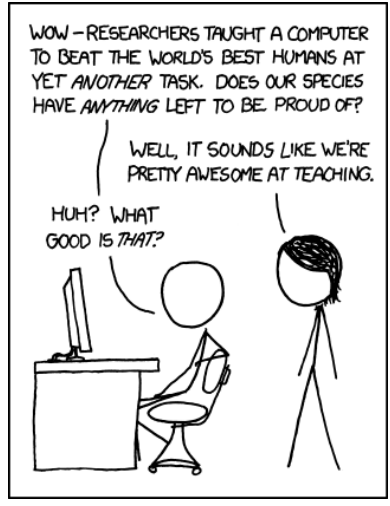


image source: xkcd.com

3

Reinforcement Learning: Deep Reinforcement Learning

Scaling up reinforcement learning

- Tabular reinforcement learning is restricted
  - small state-action spaces
  - curse of dimensionality

Scaling up reinforcement learning

- Tabular reinforcement learning is restricted
  - small state-action spaces
  - curse of dimensionality
- How can we learn to e.g. play ATARI games
  - no model available for planning
  - image is high dimensional input
  - no feasible discretization

image source: wikipedia.com

Scaling up reinforcement learning

- Tabular reinforcement learning is restricted
  - small state-action spaces
  - curse of dimensionality
- How can we learn to e.g. play ATARI games
  - no model available for planning
  - image is high dimensional input
  - no feasible discretization
- Can we replace the Q-table with a neural network?

image source: wikipedia.com

Deep Q-networks (DQN)

- Convolutional neural network $q_\theta(\boldsymbol{s}) \in \mathbb{R}^{|\mathcal{A}|}$ with one output per action

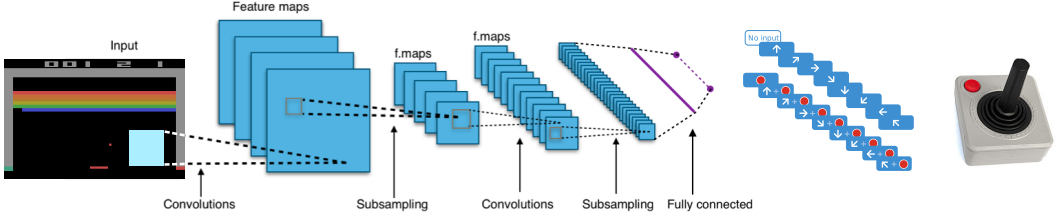
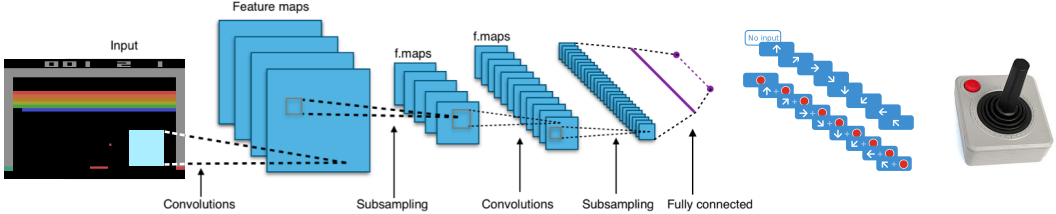


image modified from Apex34 and Bilby (wikipedia.com)

Deep Q-networks (DQN)

- Convolutional neural network $q_\theta(\boldsymbol{s}) \in \mathbb{R}^{|\mathcal{A}|}$ with one output per action



- For transitions $s, a \rightarrow r, s'$ we update paramteres $\boldsymbol{\theta}$
  - gradient descent on mean-squared TD error

$$\boldsymbol{\theta} \quad \leftarrow \quad \boldsymbol{\theta} \; - \; \alpha \nabla_\theta \underbrace{\left(r + \gamma \max_{a'} \left(q_\theta(\boldsymbol{s}')\right)_{a'} - \left(q_\theta(\boldsymbol{s})\right)_a \right)^2}_{\text{TD-error}}$$

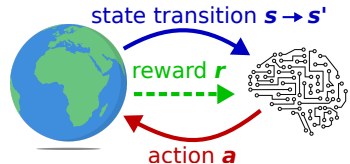
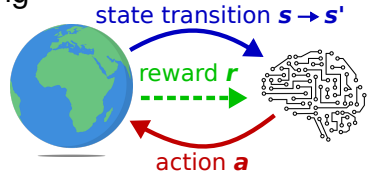


image modified from Apex34 and Bilby (wikipedia.com)

RL is not supervised learning!

- Q-learning violates assumptions of supervised learning
  - TD-error: regression targets are *non-stationary*
  - exploration: the state-distribution is *non-stationary*
  - sampling: transitions are not visited *i.i.d.*



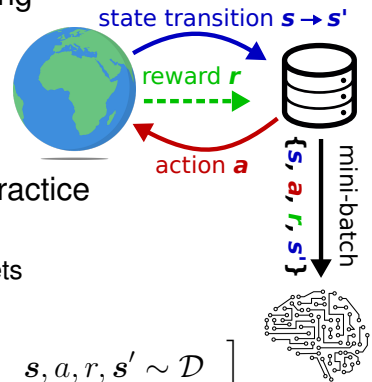
$$\boldsymbol{\theta} \quad \leftarrow \quad \boldsymbol{\theta} \; - \; \alpha \, \nabla_\theta \Big(\overbrace{r + \gamma \max_{a'} \big(q_\theta(\boldsymbol{s}')\big)_{a'}}^{\text{regression target}} - \big(q_\theta(\boldsymbol{s})\big)_a \Big)^2$$

TD-error

see Mnih et al. (2013, 2015) for details on stabilization techniques; see Badia et al. (2020) for a modern version; image modified from Designmodo

RL is not supervised learning!

- Q-learning violates assumptions of supervised learning
  - TD-error: regression targets are *non-stationary*
  - exploration: the state-distribution is *non-stationary*
  - sampling: transitions are not visited *i.i.d.*
- DQN must be stabilized with many tricks to work in practice
  - draw mini-batches i.i.d. from *experience replay buffer* $\mathcal{D}$
  - use slowly changing *target network* $q_{\theta'}$ for regression targets



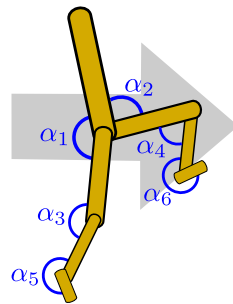
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_\theta \mathbb{E}\Big[\underbrace{\Big(r + \gamma \max_{a'} \big(\overbrace{q_{\theta'}(\boldsymbol{s}')}^{\text{target network}} \big)_{a'} - \big(q_\theta(\boldsymbol{s}) \big)_a \Big)^2}_{\text{TD error}} \Big| \underbrace{\boldsymbol{s}, a, r, \boldsymbol{s}' \sim \mathcal{D}}_{\text{from experience replay buffer}} \Big]$$

see Mnih et al. (2013, 2015) for details on stabilization techniques; see Badia et al. (2020) for a modern version; image modified from Designmodo

Continuous actions

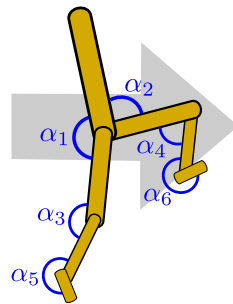
- Applications like robotics have continuous actions
  - states are limb angles, momentum, pressure sensors, etc.
    e.g. $\boldsymbol{s} = [\alpha_1, \dot{\alpha}_1, \ldots, \alpha_6, \dot{\alpha}_6]$
  - actions are motor torques, e.g. $\boldsymbol{a} = [\ddot{\alpha}_1, \ldots, \ddot{\alpha}_6] \in \mathbb{R}^{|\boldsymbol{a}|}$

$$\max_{\boldsymbol{a}} q_\theta(\boldsymbol{s}, \boldsymbol{a}) \quad = \quad ???$$



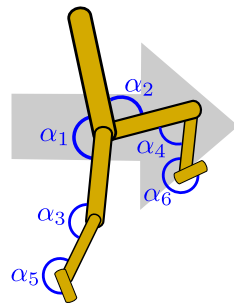
Continuous actions

- Applications like robotics have continuous actions
  - states are limb angles, momentum, pressure sensors, etc.
    e.g. $\boldsymbol{s} = [\alpha_1, \dot{\alpha}_1, \ldots, \alpha_6, \dot{\alpha}_6]$
  - actions are motor torques, e.g. $\boldsymbol{a} = [\ddot{\alpha}_1, \ldots, \ddot{\alpha}_6] \in \mathbb{R}^{|\boldsymbol{a}|}$
- Neural network outputs distribution over actions
  - e.g. a Gaussian distribution $\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) := \mathcal{N}\big(\boldsymbol{a}|\mu_\theta(\boldsymbol{s}), \sigma^2_\theta(\boldsymbol{s})\big)$
  - $\mu_\theta(\boldsymbol{s}) \in \mathbb{R}^{|\boldsymbol{a}|}$ and $\sigma_\theta(\boldsymbol{s}) \in \mathbb{R}^{|\boldsymbol{a}|}$ are neural networks



Continuous actions

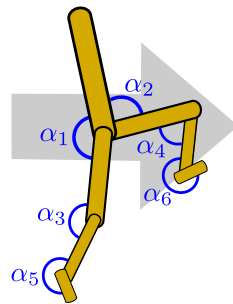
- Applications like robotics have continuous actions
  - states are limb angles, momentum, pressure sensors, etc.
    e.g. $\boldsymbol{s} = [\alpha_1, \dot{\alpha}_1, \ldots, \alpha_6, \dot{\alpha}_6]$
  - actions are motor torques, e.g. $\boldsymbol{a} = [\ddot{\alpha}_1, \ldots, \ddot{\alpha}_6] \in \mathbb{R}^{|\boldsymbol{a}|}$
- Neural network outputs distribution over actions
  - e.g. a Gaussian distribution $\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) := \mathcal{N}\big(\boldsymbol{a}|\mu_\theta(\boldsymbol{s}), \sigma^2_\theta(\boldsymbol{s})\big)$
  - $\mu_\theta(\boldsymbol{s}) \in \mathbb{R}^{|\boldsymbol{a}|}$ and $\sigma_\theta(\boldsymbol{s}) \in \mathbb{R}^{|\boldsymbol{a}|}$ are neural networks
- How can we learn the parameters θ of the policy?



Policy gradients

- Directly maximize expected return $R_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
  - starting a the first state
  - gradient reinforce actions that led to good returns

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R_0] \quad = \quad \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_t \, \nabla_\theta \ln \pi_\theta(a_t|s_t)\right]$$



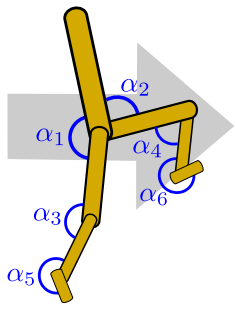
see the *policy gradient theorem* (originally by Sutton et al., 1999) in Section 13.2 of Sutton and Barto (2018)

Policy gradients

- Directly maximize expected return $R_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
  - starting a the first state
  - gradient reinforce actions that led to good returns

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R_0] \quad = \quad \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_t \, \nabla_\theta \ln \pi_\theta(a_t|s_t)\right]$$

- REINFORCE algorithm (Williams, 1992)
  - sample n trajectories per policy gradient
  - slow and not very stable



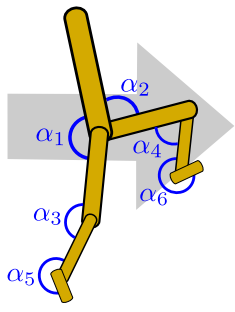
Policy gradients

- Directly maximize expected return $R_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
  - starting a the first state
  - gradient reinforce actions that led to good returns

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R_0] \quad = \quad \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_t \, \nabla_\theta \ln \pi_\theta(a_t|s_t)\right]$$

- REINFORCE algorithm (Williams, 1992)
  - sample n trajectories per policy gradient
  - slow and not very stable

- Modern variations faster and more stable
  - PPO (Schulman et al., 2017)
  - SAC (Haarnoja et al., 2018a,b)



Deep reinforcement learning

- In this video we have:
  - discussed why tabular RL does not scale
  - seen that Q-learning can be scaled up to play video games
  - learned how neural networks represent continuous policies
  - seen how continuous policies can be learned

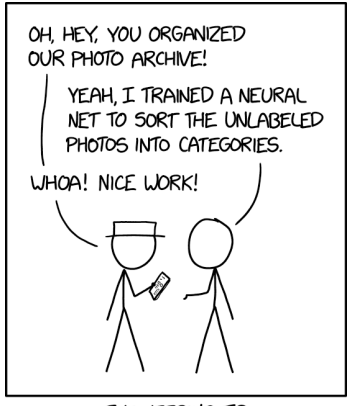


image source: xkcd.com

Deep reinforcement learning

- In this video we have:
  - discussed why tabular RL does not scale
  - seen that Q-learning can be scaled up to play video games
  - learned how neural networks represent continuous policies
  - seen how continuous policies can be learned
- But be careful, RL is not supervised learning!
- Deep RL is an active research topic

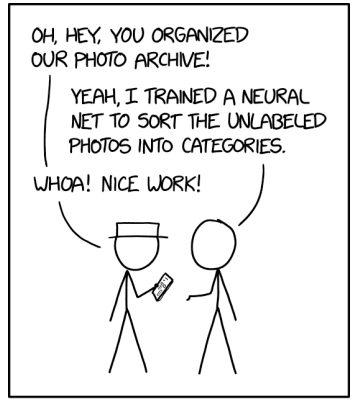


image source: xkcd.com

4

Reinforcement Learning: State-of-the-Art and Limits

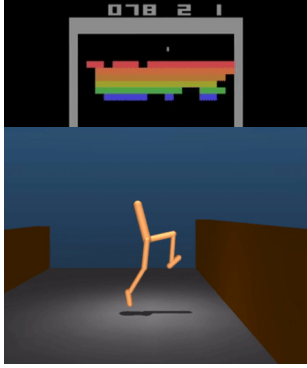
Recent RL success stories

- Playing video games (2013/2015)
  - *challenge:* high dimensional pixel input
  - one algorithm can learn each Atari game

see Mnih et al. (2013, 2015); image source: youtube.com

Recent RL success stories

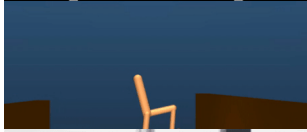
- Playing video games (2013/2015)
- Fast moving simulated robots (2016/2017)
  - *challenge:* continuous actions with complex dynamics
  - running robots pass random obstacle courses
  - only in simulators so far

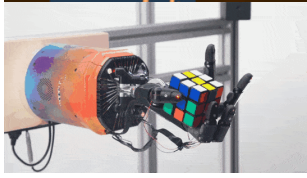


see Heess et al. (2017); image source: deepmind.com

Recent RL success stories

- Playing video games (2013/2015)
- Fast moving simulated robots (2016/2017)
- Real robotic hand solves Rubik's cubes (2019)
  - *challenge:* transfer from simulator to reality
  - works only in the lab so far

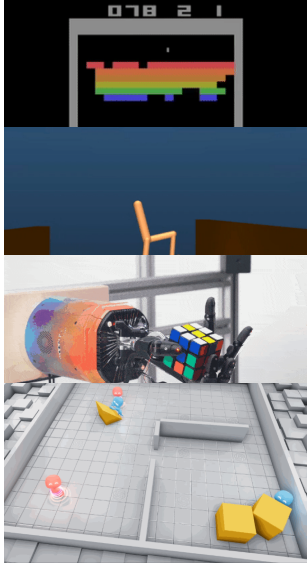




see OpenAI et al. (2019); image source: openai.com

Recent RL success stories

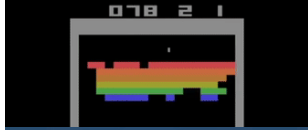
- Playing video games (2013/2015)
- Fast moving simulated robots (2016/2017)
- Real robotic hand solves Rubik's cubes (2019)
- Multi-agent RL in competitive games (2019/2020)
  - *challenge:* many agents interact with each other
  - surprisingly complex and seemingly intelligent behavior
  - RL beats professional players in StarCraft and MOBA's

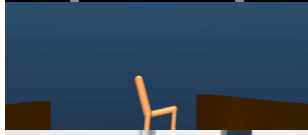


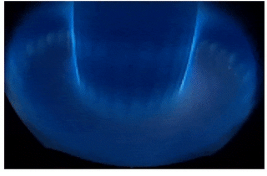
see Vinyals et al. (2019, StarCraft), Berner et al. (2019, Dota) and Baker et al. (2020, hide and seek); image source: openai.com

Recent RL success stories

- Playing video games (2013/2015)
- Fast moving simulated robots (2016/2017)
- Real robotic hand solves Rubik's cubes (2019)
- Multi-agent RL in competitive games (2019/2020)
- Magnetic control of a Tokamak fusion reactor (2022)
  - *challenge:* keep the plasma bubble from bursting
  - simulations transferred to real fusion reactor



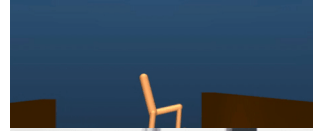




see Degrave et al. (2022); image source: deepmind.com

Recent RL success stories

- Playing video games (2013/2015)
- Fast moving simulated robots (2016/2017)
- Real robotic hand solves Rubik's cubes (2019)
- Multi-agent RL in competitive games (2019/2020)
- Magnetic control of a Tokamak fusion reactor (2022)
- Most experts expect breakthrough in RL soon
  - industry applications within a decade or so
  - might impact **your** future job!



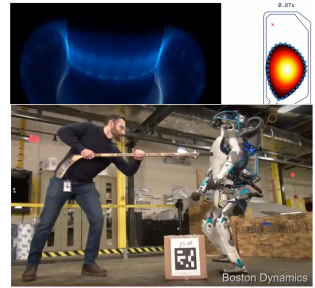


What is possible?

- Now: learn to control unknown dynamics
  - environment must be stationary and completely explorable
  - many interactions with the environment (e.g. simulator/game)
  - no catastrophic/irreversible actions (e.g. crashing the car)

image source: Tokamak fusion reactor from deepmind.com

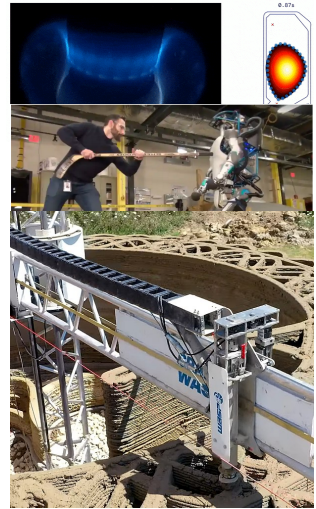
What is possible?



- Now: learn to control unknown dynamics
  - environment must be stationary and completely explorable
  - many interactions with the environment (e.g. simulator/game)
  - no catastrophic/irreversible actions (e.g. crashing the car)
- Soon: safe transfer of learned control
  - learn in a simulation, act safely in the real world
  - learn to anticipate behavior of other learning agents
  - integrate existing knowledge/constraints into the training

image source: robot stability test from Boston Dynamics

What is possible?



- Now: learn to control unknown dynamics
  - environment must be stationary and completely explorable
  - many interactions with the environment (e.g. simulator/game)
  - no catastrophic/irreversible actions (e.g. crashing the car)
- Soon: safe transfer of learned control
  - learn in a simulation, act safely in the real world
  - learn to anticipate behavior of other learning agents
  - integrate existing knowledge/constraints into the training
- Possible applications
  - infinite sampling: computer games, web-crawling, video processing
  - limited autonomy: construction, factories, power plants/grids, driver assistance

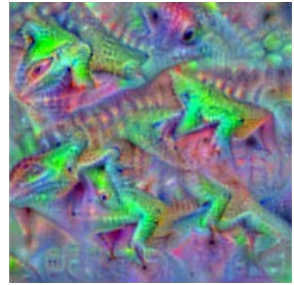
image source: 3D-printed construction by Alfredo Milano (from wikipedia)

What is impossible?

- RL agents do **not represent** the world as we do
  - only patterns: no objects, relationships, movements
  - only training data: no intuition of what is *similar*

What you see:

What a neural net sees:



visualizing “archetypical inputs” for neural nets; image source: Girish Gowda and Farhad Dalirani

What is impossible?

- RL agents do **not represent** the world as we do
  - only patterns: no objects, relationships, movements
  - only training data: no intuition of what is *similar*
- RL agents do **not anticipate** new situations
  - reinforce actions: all situations have to be trained for
  - no imagination: all reaction, no prediction

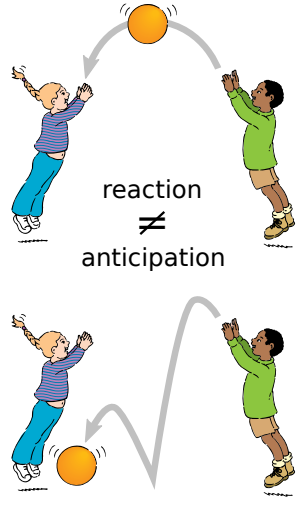
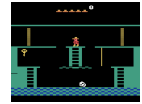


image source: modified from freesvg.org

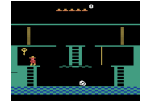
What is impossible?

- RL agents do **not represent** the world as we do
  - only patterns: no objects, relationships, movements
  - only training data: no intuition of what is *similar*
- RL agents do **not anticipate** new situations
  - reinforce actions: all situations have to be trained for
  - no imagination: all reaction, no prediction
- RL agents **only learn** the task they are given
  - only looks smart: memorizes what yielded reward
  - misspecification: RL can act dumb to get reward

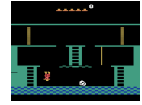
1. start



2. go get the key

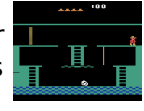


3. **die**



4. respawn near door

5. open door get points



example of the Atari game *Montezuma's Revenge*, available in OpenAI's `gym library`

What is impossible?

- RL agents do **not represent** the world as we do
  - only patterns: no objects, relationships, movements
  - only training data: no intuition of what is *similar*
- RL agents do **not anticipate** new situations
  - reinforce actions: all situations have to be trained for
  - no imagination: all reaction, no prediction
- RL agents **only learn** the task they are given
  - only looks smart: memorizes what yielded reward
  - misspecification: RL can act dumb to get reward

- RL agents are **not intelligent** like humans! Don't expect them to be!

image source: *The Simpsons* (20th Television Animation)

When to use reinforcement learning?

1. Is there no heuristic that already solves the task almost optimal?
2. Is your task virtual or do you have a simulator of your tasks?
3. Are there no “forbidden” actions that would destroy the system?
4. Can the system cause **no** serious harm to people?
5. Is the variety of situations the agent can face limited?
6. Do you have a lot of compute to train the agent for all situations?
7. Are you willing to spend 6 month fiddling with hyper-parameters?

State-of-the-art and limits

- In this video we have:
  - seen some recent success stories of RL
  - discussed what RL *can* learn
  - discussed what RL *cannot* learn
  - discussed when to use RL

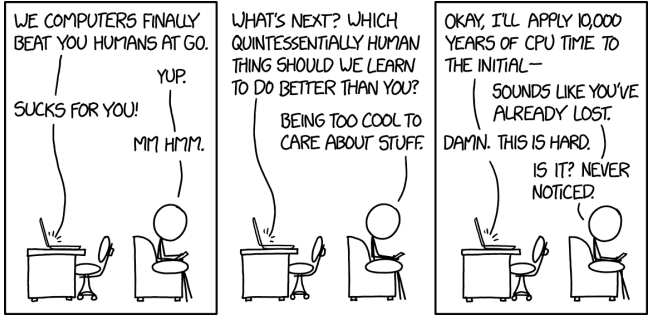


image source: xkcd.com

References I

Adriá Puigdoménech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari human benchmark. In *CoRR, abs/2003.13350*, 2020. URL https://arxiv.org/abs/2003.13350.

Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *8th International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=SkxpxJBKwS.

Richard Ernest Bellman. *Dynamic programming*. Princeton University Press, 1957.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602:414–419, 02 2022. doi: 10.1038/s41586-021-04301-9.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of Machine Learning Research (ICML)*, volume 80, pages 1861–1870, 2018a. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018b. URL http://arxiv.org/abs/1812.05905.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL http://arxiv.org/abs/1707.02286.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602. NIPS Deep Learning Workshop 2013.

References II

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019. URL http://arxiv.org/abs/1910.07113.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Richard S. Sutton, David McAllester, and Satinder Singh anf Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 12, pages 1057–1063. MIT Press, 1999. URL https://papers.nips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019.

Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Ronald J. Williams. Simple statistical gradient algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. URL https://link.springer.com/article/10.1007/BF00992696.