Introduction to Reinforcement Learning

Machine Learning

András Attila Sulyok

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

21/05/2024

What is the goal of this lecture?

- Understand the formulation in RL
  When can it be used?
  Why is it difficult?

- Understand the idea behind behind REINFORCE and Q-learning
  Two different approaches

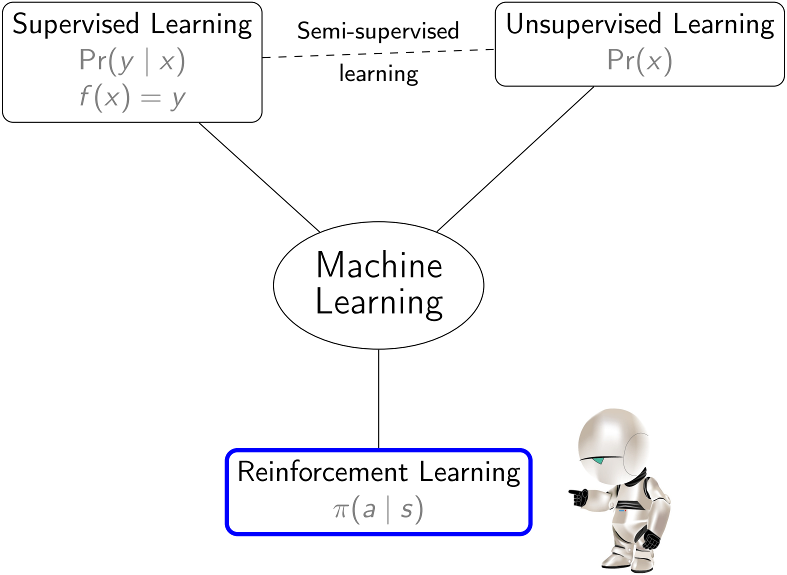
Fun Time

How familiar are you with RL?

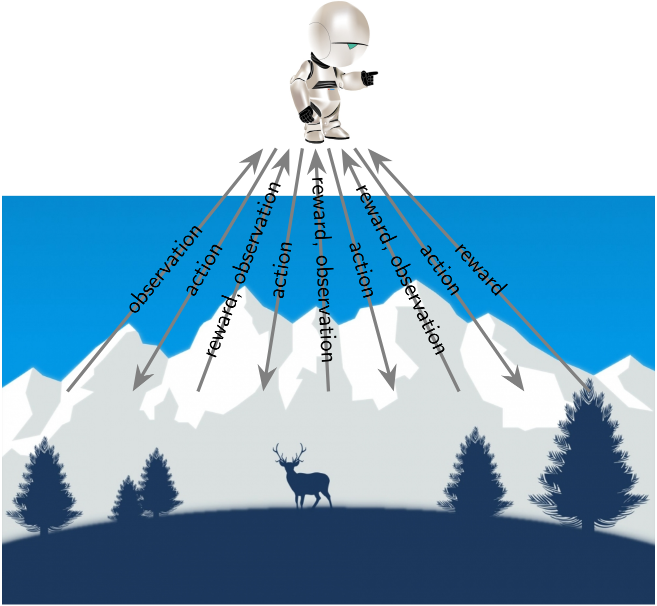
1. With the what?
2. I know about the theory.
3. I have implemented some RL algorithms.
4. I contribute new theory to the community.
5. I have my own army of cyber-lizardpeople AGIs ready to take on the world.

Reference answer: None this time.

Branches of Machine Learning

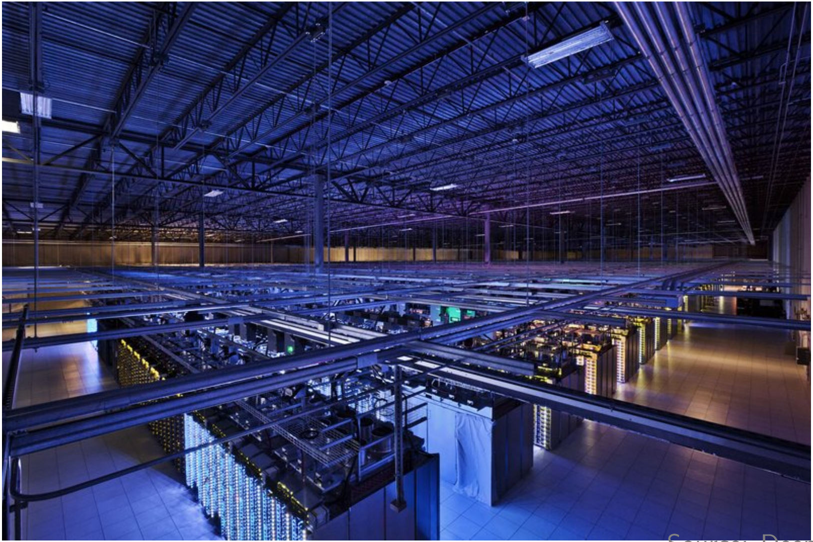


Reinforcement Learning



Applications

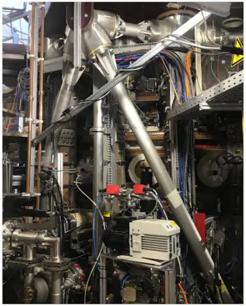
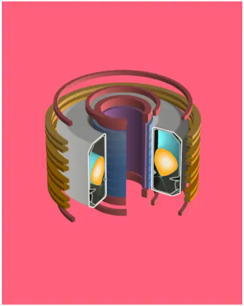
Cooling and resource management systems



Source: DeepMind

Applications

Nuclear fusion control


https://www.deepmind.com/blog/article/Accelerating-fusion-science-through-learned-plasma-control

Source: DeepMind

Multi-armed Bandit

Source: offtopia.net

Formally

There are k discrete *actions*: $a \in \mathcal{A}$
There is a reward associated with each action: $R(a)$
this is a random variable

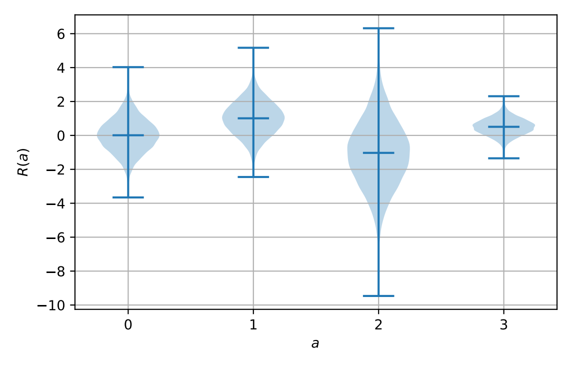
Goal: give actions $a_1, a_2, \dots$ so that the cumulative reward is maximal:

$$\arg\max_{a_1, a_2, \dots} \mathbb{E}\left\{ \sum_{t=1}^{\infty} R(a_t) \right\}$$

Catch: you don't know R!
You can only observe the realizations

Fun Time

Which action would you choose?



Reference answer: 1

Demo for home: `https://iosband.github.io/2015/07/28/Beat-the-bandit.html`

How to solve this?

Value estimation

Define the value of an action:

$$Q(a) = \mathbb{E}\{R(a)\}$$

Estimate it by averaging:

$$Q(a) \approx q(a) = \frac{1}{T}\sum_{t=1}^{T} r_t,$$

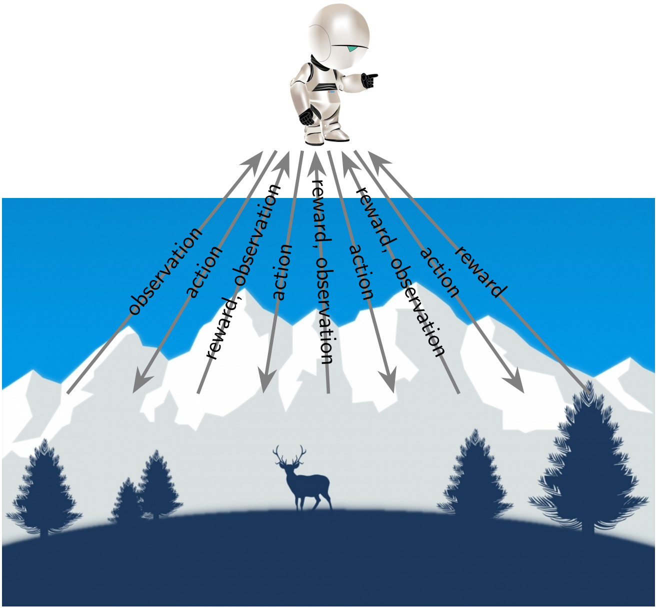
where r_t are the concrete realisations (sampled according to the distribution of $R(a)$).

Then simply choose $\arg\max_a q(a)$.

That's what you call interaction?

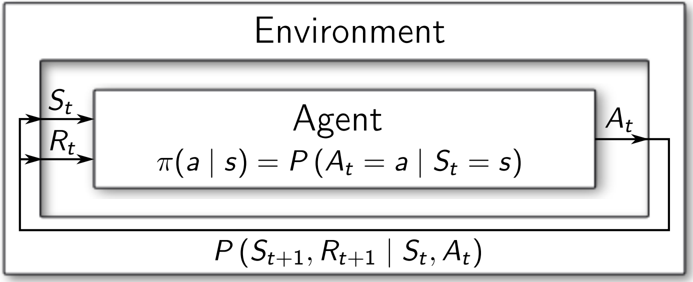
Source: integratingtech301.pbworks.com

No, this is interaction



General formulation

with a figure



Markov-property: next state depends only on current state $\implies$ probability of the whole trajectory is a product of probs. of transitions

Markov Decision Process

Let's say we have an environment with states $s \in \mathcal{S}$ and posssible actions $a \in \mathcal{A}$.

The dynamics $\mathcal{P}$ gives the probabilities of the next state s_{t+1} given the current state s_t and the given action a_t.

The reward function $R(s, a)$ gives the expected reward for action a given in state s.

The policy $\pi(a \mid s)$ of the agent is the probability of taking action a in state s.
Abuse of notation: *deterministic policy* is denoted by $a = \pi(s)$ (ie. taking action a w. p. 1).

Expected (undiscounted) return:

$$G^\pi(s) = \mathbb{E}_{a_t \sim \pi} \mathbb{E}_{s_t \sim \mathcal{P}} \left\{ \sum_{t=0}^{T} R(s_t, a_t) \right\}$$

Goal: find a policy $\pi = \arg\max_\pi G^\pi(s_0)$ (for some initial state s_0).

Fun Time

What is the *minimal* expected return in the following environment?

- From Start to Goal
- Reward for each step is -0.01
- Reward for stepping onto water (blue): -0.1
- Reward for hitting a wall: -0.5
- Reward for stepping on goal: $+1$

1. 0
2. 0.33
3. 0.83
4. 0.93
5. 1

Reference answer: 4

$$7 \cdot (-0.01) + 1 = 0.93$$

Stochastic Gradient Ascent

Objective: maximise

$$G^\pi = \mathbb{E}_{a_t \sim \pi, s_t, r_t \sim \mathcal{P}} \left\{ \sum_{t=1}^{T} r_t \right\}$$

Idea: $\pi = \pi_\theta$ with some parameter θ, and do gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta G^\pi$$

Calculating the gradient

Let $\tau = (s_0, a_0, s_1, \dots, s_T)$ the trajectory.

$$\begin{aligned}
\nabla_\theta G^\pi &= \nabla_\theta \mathbb{E}_{a_t \sim \pi_\theta} \mathbb{E}_{s_t \sim \mathcal{P}} R(\tau) \\
&= \mathbb{E}_{a_t \sim \pi_\theta} \mathbb{E}_{s_t \sim \mathcal{P}} \nabla_\theta \log P_\theta(\tau) R(\tau) \\
&= \mathbb{E}_{a_t} \mathbb{E}_{s_t} \left\{ \nabla_\theta \log \prod_t \pi_\theta(a_t \mid s_t) P(s_{t+1} \mid s_t, a_t) R(\tau) \right\} \\
&= \mathbb{E}_{a_t} \mathbb{E}_{s_t} \left\{ \nabla_\theta \left[\sum_t \log \pi_\theta(a_t \mid s_t) + \log P(s_{t+1} \mid s_t, a_t) \right] R(\tau) \right\} \\
&= \mathbb{E}_{a_t} \mathbb{E}_{s_t} \left\{ \left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] R(\tau) \right\}
\end{aligned}$$

A Policy Gradient algorithm

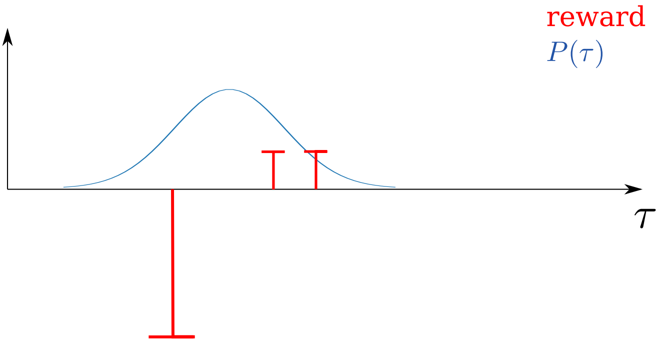
$$\theta \leftarrow \theta + \alpha \nabla_\theta G^\pi$$

$$\nabla_\theta G^\phi = \mathbb{E}_{a_t} \mathbb{E}_{s_t} \left\{ \left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] r(\tau) \right\}$$

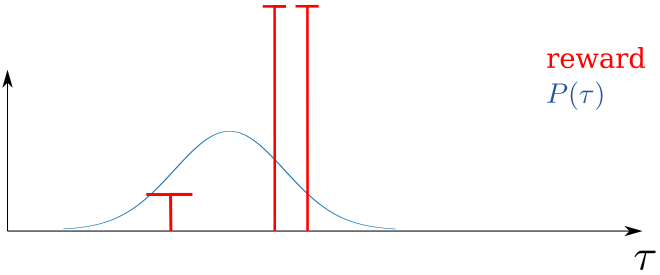
The REINFORCE algorithm:

1. run your agent (π_θ), sample some trajectories $\{\tau^{(i)}\}$
2. estimate $G^\pi \approx \frac{1}{N} \sum_i \hat{G}^\pi(\tau^{(i)})$ (by the return)
3. estimate the gradient
4. update: $\theta \leftarrow \theta + \alpha \nabla_\theta \hat{G}^\pi$
5. repeat

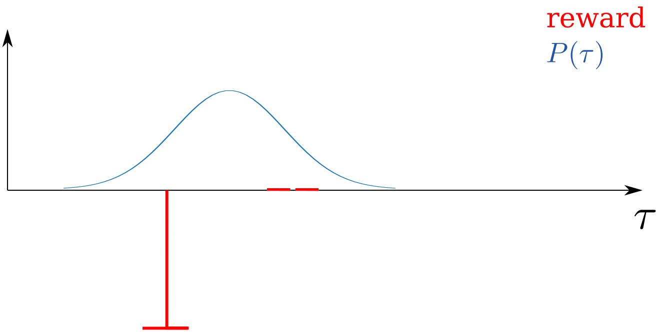
What's wrong with PG?



What's wrong with PG?



What's wrong with PG?



Variance reduction

by exploiting causality

Problem: high variance of the gradient estimation

But the policy at time t can't affect reward at time $t' < t$:

$$\nabla_\theta G^\phi = \mathbb{E}_{a_t}\mathbb{E}_{s_t}\left\{\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\sum_{t'=t}^{T} R(s_t, a_t)\right)\right]\right\}$$

Variance reduction

by subtracting a baseline

Have a look at this:

$$\mathbb{E}_{a_t}\mathbb{E}_{s_t}\left\{\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right](r(\tau) - b)\right\}$$

for some $b \in \mathbb{R}$.

$$\mathbb{E}\{\nabla_\theta \log P_\theta(\tau) b\} = b\mathbb{E}\{\nabla_\theta \log P_\theta(\tau)\} = b\nabla_\theta \mathbb{E}\{1\} = b\nabla_\theta 1 = 0$$

So this is also an unbiased estimator, but the variance is different!
$b = \overline{R} = \frac{1}{N}\sum_i r_i$ works in practice and is easy to compute.

How good is a policy?

Remember the bandits? We wanted to choose the arm with the maximal expected reward:

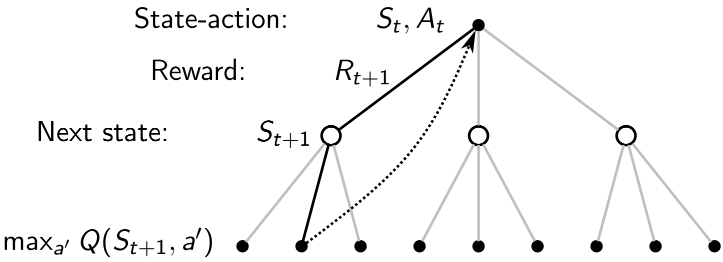
$$\arg\max_a \mathbb{E}\{R(a)\} =: Q(a)$$

Similarly in MDPs:

$$Q(s,a) := \mathbb{E}\{\sum_{t=1}^{T} R(s_t, a_t) \mid s_0 = s, a_0 = a, \pi\}$$

the *value of the state and action* for a policy π

Graphical intuition



Temporal Difference *(TD) learning*: based on the difference of the value function in two consecutive timesteps.

Bellman equation

We want to estimate how well the policy performs:

$$Q(s, a) = R(s, a) + \mathbb{E}_{s' \sim \mathcal{P}} \mathbb{E}_{a' \sim \pi} \{ Q(s', a') \}$$

$(\forall s, a)$ This is the Bellman equation.

Well, let's update during interaction: at a specific step t:

$$Q(s_t, a_t) \leftarrow r_t + Q(s_{t+1}, \pi(s_{t+1}))$$

Do this many times, and Q will converge for a policy.

Policy improvement

So far, the policy π stayed fixed. How to improve?

Idea: Greedy policy wrt Q:

$$\pi(s) = \arg\max_a Q(s, a)$$

Note: *deterministic policy*

Provably not worse

Generalized Policy Iteration

Putting it together:

- estimate Q for a given π
- improve π
  $\pi(s) \leftarrow \arg\max_a Q(s, a)$
- repeat

A bit slow...

Idea #2: do these two in parallel: *Generalized Policy Iteration*

$$Q(s_t, a_t) \leftarrow r_t + \max_{a'} Q(s_{t+1}, a')$$

Q-learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Will provably converge under some assumptions.
Note that it (as is) works only when $|\mathcal{S}| < \infty$.

Source: Sutton and Barto [1]

State of the art algorithms

- Policy Gradient → Proximal Policy Optimization (PPO)
  - ChatGPT fine-tuning, Dota
- Q learning → Deep Q Network (DQN) → Rainbow → Agent57
  - Atari games
- Maximum a Posteriori Policy Optimization (MPO)
  - "Suppose the agent will perform well, what is the probability for this action?"
  - AlphaTensor, plasma controller
- Monte Carlo Tree Search (MTCS) → AlphaGo 0 → MuZero
  - Assumes known dynamics; similar to planning
  - Go, Chess, Atari games
- PG + Q learning → Actor Critic → IMPALA
  - AlphaStar

Further reading

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html (visited on 05/12/2018).

[2] C. Szepesvári, "Reinforcement learning algorithms for MDPs," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[3] Y. Li, "Deep reinforcement learning,", Oct. 15, 2018. arXiv: http://arxiv.org/abs/1810.06339v1 [cs.LG].

- DRL course at UC Berkeley
- Spinning Up in Deep RL
- EEML RL tutorials: 2021, 2022