UNIVERSITY OF COPENHAGEN

Reinforcement Learning (RL)

MLS 2025, Data Science Lab, UCPH

Tong Chen (toch@di.ku.dk)

Postdoc
Dept. of Computer Science (ML Section)
University of Copenhagen

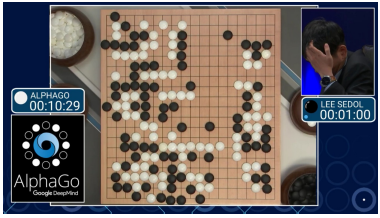
With Raghav (raghav@di.ku.dk)

November 27, 2025

Overview

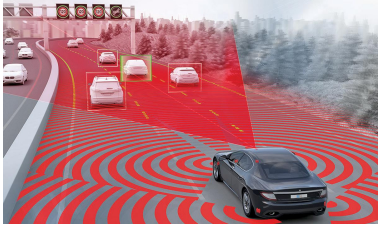
- General Introduction
- Dynamic Programming
- Deep Q-Learning

RL is Everywhere

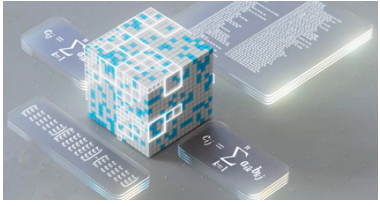


AlphaGo

AlphaStar

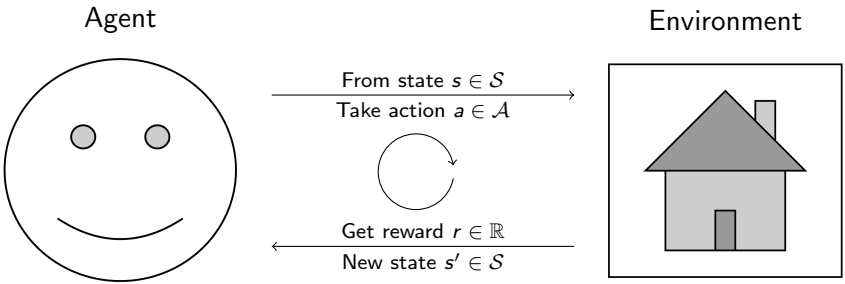


Autonomous Driving

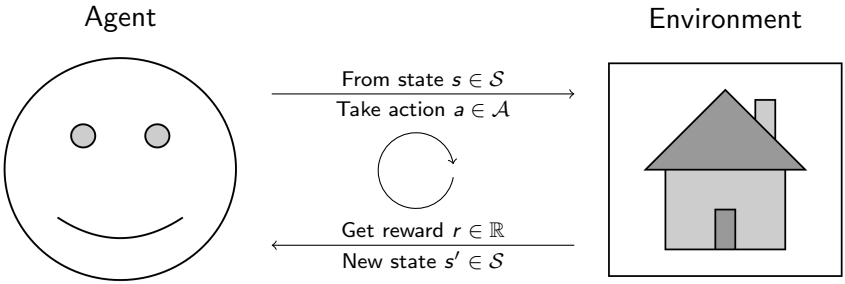


AlphaTensor

RL: Interacting with the Environment



Formalizing decision process: Markov Decision Process (MDP)



- Markov property:

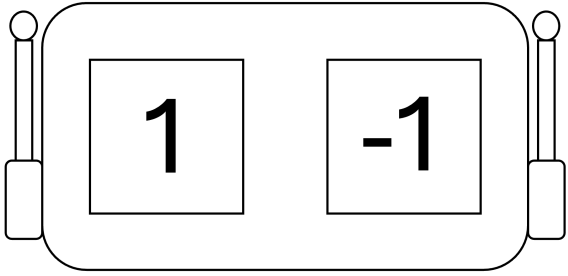
$$\mathbf{Pr}[S_{t+1}|S_t] = \mathbf{Pr}[S_{t+1}|S_1, \ldots S_t];$$

- Goal: Learn an optimal strategy (policy) $\pi : \mathcal{S} \to \mathcal{A}$.

Find Your Own RL Example

- Scenario: Greenhouse environment control;
- State: Plants, environment condition (light, humidity, temperature, etc.);
- Action: Change of environment condition;
- Reward: Positive (growing fast, thrive) / Negative (growing slow, withered);
- Horizon: Finite (vegetables, flowers) / Infinite (trees).

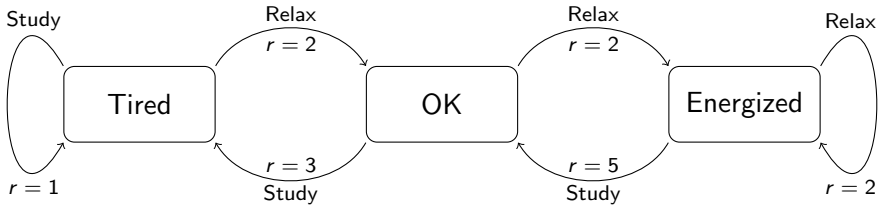
Example: Two-Armed Bandit



How to guess the arm with high probability?

Example: Plan Your Study

- States: "Tired", "OK", "Energized";
- Actions: "Relax", "Study";
- Transition: Day-by-day and deterministic.



What is your learning strategy?

Dynamic Programming (DP): Finite Horizon ($n = 1$)

$$Q_1(s,a) = R(s,a),\ V_1(s) = \max_a Q_1(s,a),\ \pi_1(s) = \arg\max_a Q_1(s,a)$$

Dynamic Programming (DP): Finite Horizon ($n = 1$)

$$Q_1(s,a) = R(s,a),\ V_1(s) = \max_a Q_1(s,a),\ \pi_1(s) = \arg\max_a Q_1(s,a)$$

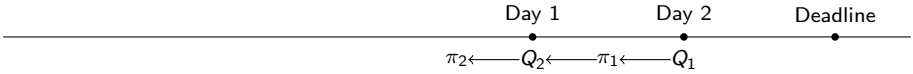
$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	**2**	1
O	2	**3**
E	2	**5**

Q-value $Q_1(s,a)$

State	$V_1(s)$	$\pi_1(s)$
T	2	Relax
O	3	Study
E	5	Study

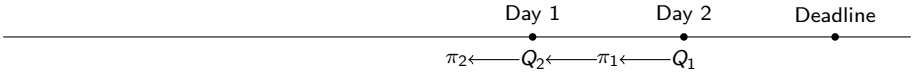
V-value $V_1(s)$ and policy $\pi_1(s)$

Dynamic Programming (DP): Finite Horizon ($n = 2$)



$$Q_2(s,a) = R(s,a) + \max_{a'} Q_1(s',a'),\ V_2(s) = \max_{a} Q_2(s,a),\ \pi_2(s) = \arg\max_{a} Q_2(s,a)$$

Dynamic Programming (DP): Finite Horizon ($n = 2$)



$$Q_2(s,a) = R(s,a) + \max_{a'} Q_1(s',a'),\ V_2(s) = \max_{a} Q_2(s,a),\ \pi_2(s) = \arg\max_{a} Q_2(s,a)$$

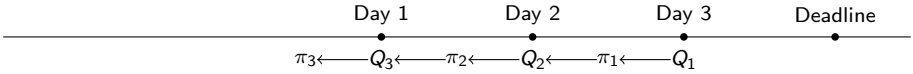
$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	**5**	3
O	**7**	5
E	7	**8**

Q-value $Q_2(s,a)$

State	$V_2(s)$	$\pi_2(s)$
T	5	Relax
O	7	Relax
E	8	Study

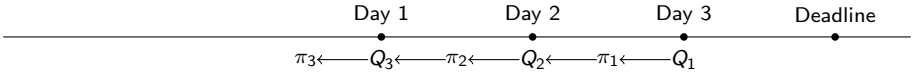
V-value $V_2(s)$ and policy $\pi_2(s)$

Dynamic Programming (DP): Finite Horizon ($n = 3$)



$$Q_3(s,a) = R(s,a) + \max_{a'} Q_2(s',a'),\ V_3(s) = \max_{a} Q_3(s,a),\ \pi_3(s) = \arg\max_{a} Q_3(s,a)$$

Dynamic Programming (DP): Finite Horizon ($n = 3$)



$$Q_3(s,a) = R(s,a) + \max_{a'} Q_2(s',a'),\ V_3(s) = \max_{a} Q_3(s,a),\ \pi_3(s) = \arg\max_{a} Q_3(s,a)$$

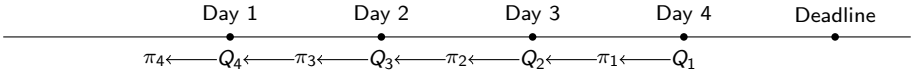
$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	**9**	6
O	**10**	8
E	10	**12**

Q-value $Q_3(s,a)$

State	$V_3(s)$	$\pi_3(s)$
T	9	Relax
O	10	Relax
E	12	Study

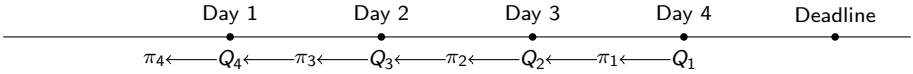
V-value $V_3(s)$ and policy $\pi_3(s)$

Dynamic Programming (DP): Finite Horizon ($n = 4$)



$$Q_4(s,a) = R(s,a) + \max_{a'} Q_3(s',a'),\ V_4(s) = \max_{a} Q_4(s,a),\ \pi_4(s) = \arg\max_{a} Q_4(s,a)$$

Dynamic Programming (DP): Finite Horizon ($n = 4$)



$$Q_4(s,a) = R(s,a) + \max_{a'} Q_3(s',a'),\ V_4(s) = \max_{a} Q_4(s,a),\ \pi_4(s) = \arg\max_{a} Q_4(s,a)$$

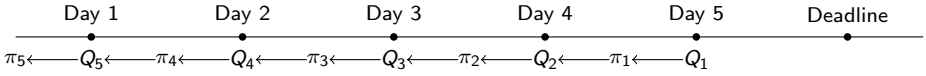
$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	**12**	10
O	**14**	12
E	14	**15**

Q-value $Q_4(s,a)$

State	$V_4(s)$	$\pi_4(s)$
T	12	Relax
O	14	Relax
E	15	Study

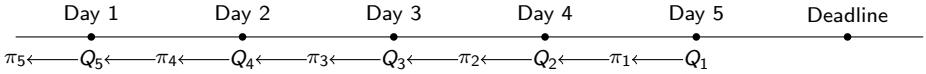
V-value $V_4(s)$ and policy $\pi_4(s)$

Dynamic Programming (DP): Finite Horizon ($n = 5$)



$$Q_5(s,a) = R(s,a) + \max_{a'} Q_4(s',a'),\ V_5(s) = \max_{a} Q_5(s,a),\ \pi_5(s) = \arg\max_{a} Q_5(s,a)$$

Dynamic Programming (DP): Finite Horizon ($n = 5$)



$$Q_5(s,a) = R(s,a) + \max_{a'} Q_4(s',a'),\ V_5(s) = \max_{a} Q_5(s,a),\ \pi_5(s) = \arg\max_{a} Q_5(s,a)$$

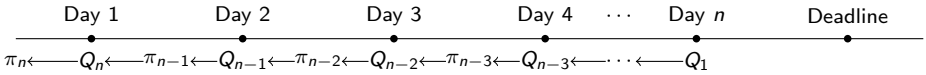
$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	**16**	13
O	**17**	15
E	17	**19**

Q-value $Q_5(s,a)$

State	$V_5(s)$	$\pi_5(s)$
T	16	Relax
O	17	Relax
E	19	Study

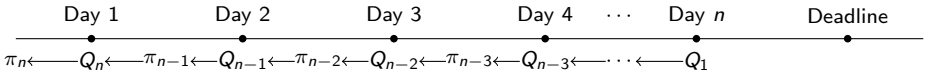
V-value $V_5(s)$ and policy $\pi_5(s)$

Dynamic Programming (DP): Infinite Horizon ($n \to \infty$)



$$Q_n(s,a) = R(s,a) + \max_{a'} Q_{n-1}(s',a'),\ V_n(s) = \max_a Q_n(s,a),\ \pi_n(s) = \arg\max_a Q_n(s,a)$$

Dynamic Programming (DP): Infinite Horizon ($n \to \infty$)



$$Q_n(s,a) = R(s,a) + \max_{a'} Q_{n-1}(s',a'),\ V_n(s) = \max_a Q_n(s,a),\ \pi_n(s) = \arg\max_a Q_n(s,a)$$

$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	∞	∞
O	∞	∞
E	∞	∞

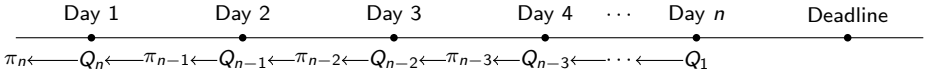
Q-value $Q_n(s,a)$

State	$V_n(s)$	$\pi_n(s)$
T	∞	?
O	∞	?
E	∞	?

V-value $V_n(s)$ and policy $\pi_n(s)$

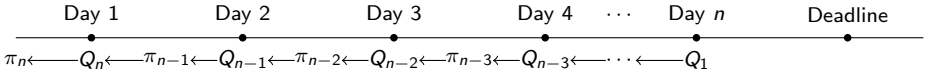
What's wrong? How to solve the issue?

Dynamic Programming (DP): Infinite Horizon ($n \to \infty, \gamma = 0.5$)



$$Q_n(s,a) = R(s,a) + \gamma \max_{a'} Q_{n-1}(s',a'),\ V_n(s) = \max_a Q_n(s,a),\ \pi_n(s) = \arg\max_a Q_n(s,a)$$

Dynamic Programming (DP): Infinite Horizon ($n \to \infty, \gamma = 0.5$)



$$Q_n(s,a) = R(s,a) + \gamma \max_{a'} Q_{n-1}(s',a'),\ V_n(s) = \max_a Q_n(s,a),\ \pi_n(s) = \arg\max_a Q_n(s,a)$$

$\mathcal{S}$ \ $\mathcal{A}$	Relax	Study
T	**5**	3.5
O	**6**	5.5
E	6	**8**

Q-value $Q_n(s,a)$

State	$V_n(s)$	$\pi_n(s)$
T	5	Relax
O	6	Relax
E	8	Study

V-value $V_n(s)$ and policy $\pi_n(s)$

Adding discounting factor $0 \leq \gamma \leq 1$ for future (long-term) reward helps!

Summary

- Three elements: state s, action a, reward r;
- Future reward at (s, a) = immediate reward at (s, a) + maximum future reward at s';
- Dynamic programming: Simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner;
- Discounting factor γ: Future rewards play a key role in present decisions, but their impacts should diminish over time;
- Goal of RL: Obtain stable (converged) Q-values and corresponding policy.

Deep Q-Learning: Learn from a Static Environment

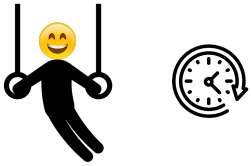
- Use neural network to approximate optimal Q-value: $Q_\theta \approx Q_*$;
- Sample (s, a, r, s') from a **static** environment and compute loss:

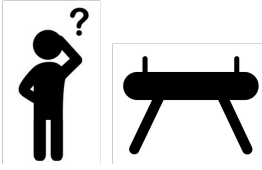
$$L_\theta = \left(Q_\theta(s, a) - \left(r + \gamma \max_{a'} Q_\theta(s', a') \right) \right)^2;$$

- Obtain optimal V-value and policy:

$$V_\theta(s) = \max_a Q_\theta(s, a),$$
$$\pi_\theta(s) = \arg\max_a Q_\theta(s, a).$$

Let's learn by doing!





Deep Q-Learning: Experience Replay

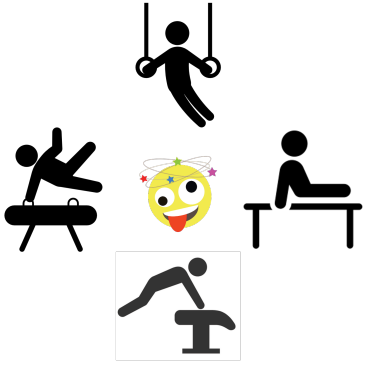
- Use neural network to approximate optimal Q-value: $Q_\theta \approx Q_*$;
- Sample $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$ from **random** environments and compute loss:

$$L_\theta = \frac{1}{T}\sum_{t=1}^{T}\left(Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_a Q_\theta(s_{t+1}, a)\right)\right)^2;$$

- Obtain optimal V-value and policy:

$$V_\theta(s) = \max_a Q_\theta(s, a),$$
$$\pi_\theta(s) = \arg\max_a Q_\theta(s, a).$$

Let's learn by doing!



Deep Q-Learning: Target Q-Network

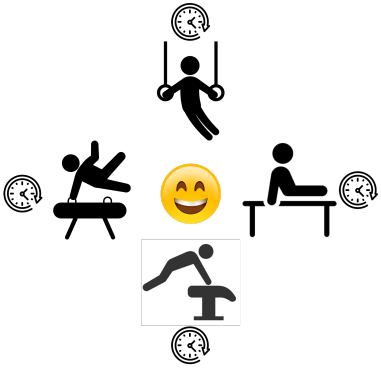
- Use **two** identical neural networks to approximate optimal Q-value: $Q_\theta = Q_\theta^{target} \approx Q_*$;
- Sample $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$ from **random** environments (using Q_θ) and compute loss:

$$L_\theta = \frac{1}{T}\sum_{t=1}^{T}\left(Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_a Q_\theta^{target}(s_{t+1}, a)\right)\right)^2;$$

- During training, we freeze Q_θ^{target} and only update Q_θ;
- But every so often, we copy weights from Q_θ to Q_θ^{target};
- Obtain optimal V-value and policy:

$$V_\theta(s) = \max_a Q_\theta(s, a),$$
$$\pi_\theta(s) = \arg\max_a Q_\theta(s, a).$$

Let's learn by doing!



Summary

- Deep Q-Learning: Using neural network to approximate optimal Q-value;
- Experience replay: Adapting policy to dynamic environments;
- Target Q-network: Stabilizing learning process.