

Approximate value-based RL

How to approximately solve an RL problem

Cathy Wu

6.7920 Reinforcement Learning: Foundations and Methods

Readings

1. *DPOC vol2 §2.5.3*
2. Neuro-Dynamic Programming (NDP). §3.1-3.2.
3. [SB 16.5](#)

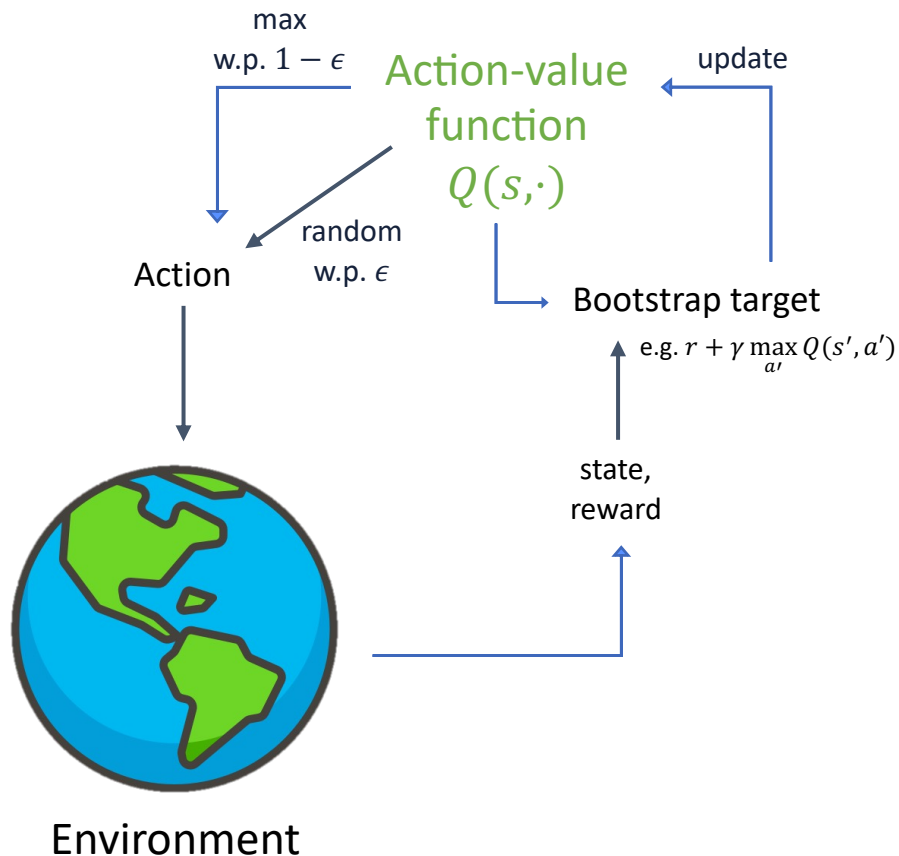
Learning milestone!

By the end of this lecture,
you should understand the foundations of Deep Q Networks (DQN),
the launching point of modern RL. Phew!

Value-based RL methods

Example:

- Q-learning
- SARSA
- Approximate value iteration
- Fitted Q-iteration
- DQN
- Double DQN
- ...



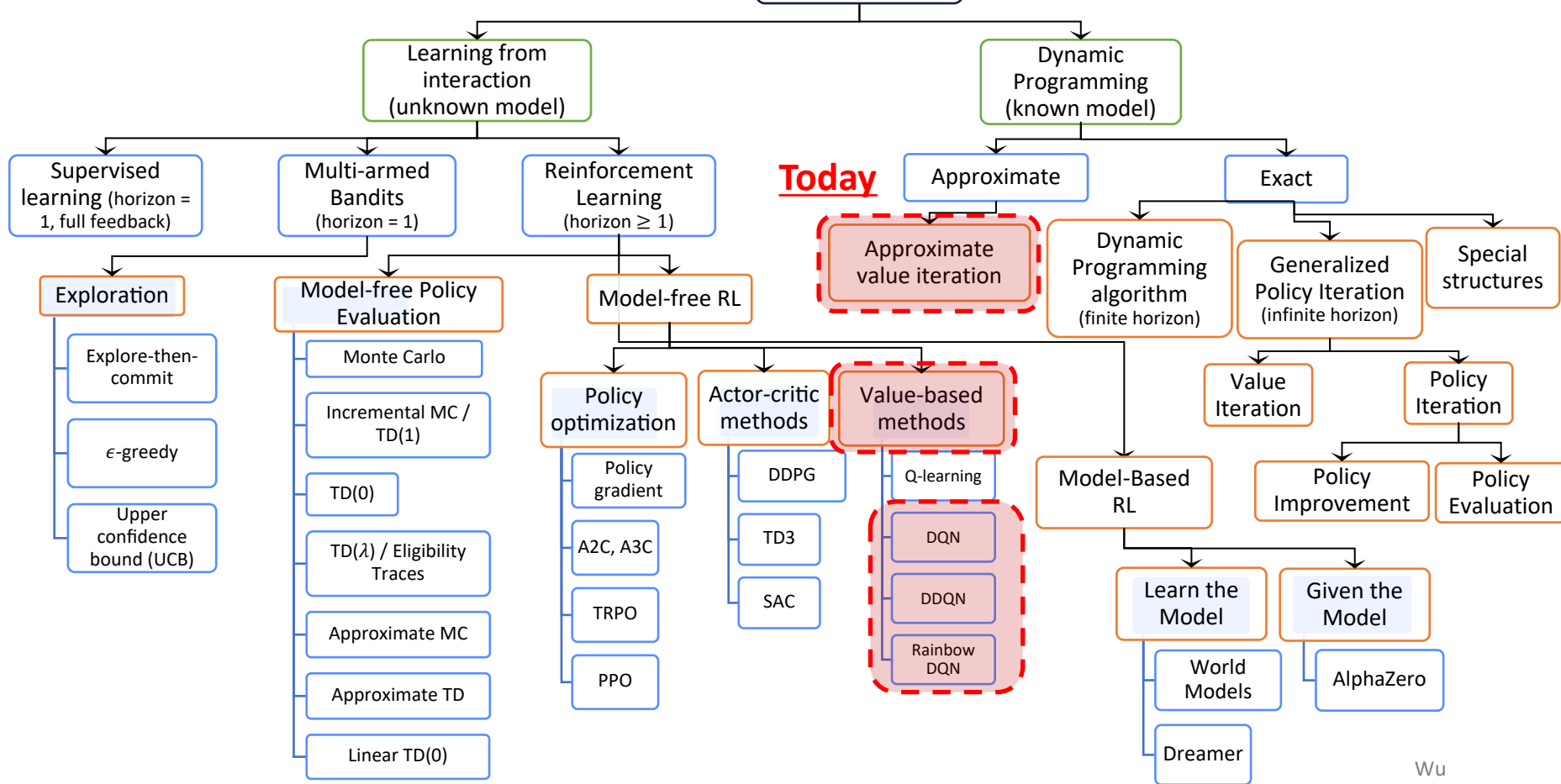
Outline

1. Approximate value-based algorithms

Taxonomy of RL Methods

Methods for
sequential
decision making

Reinforcement learning (RL) is a framework for
sequential decision making under uncertainty.



Notice

Reminder: We are typically working in the episodic discounted setting.

Most results smoothly extend to other settings.

NEW: The value functions **cannot** be represented **exactly** (as in the tabular setting).

Outline

- 1. Policy learning with approximate value functions**
 - a. Approximate value iteration
 - b. Fitted Q iteration
 - c. Deep Q-network (DQN)
 - d. Double DQN (DDQN)
 - e. Rainbow DQN
 - f. Revisiting Rainbow

Approximate value iteration (AVI)

- Recall: value iteration. Start with any V_0 . Then:

$$V_{k+1} = \mathcal{T}V_k$$

- Contraction property of $\mathcal{T} \Rightarrow V_k \rightarrow V^*$.

- **Approximate value iteration**. Start with any V_0 . Then:

$$V_{k+1} = \mathcal{A}\mathcal{T}V_k$$

where $\mathcal{A}$ is a generic approximation operator.

- Standard case:

$$V_{k+1} = \arg \inf_{V \in \mathcal{F}} ||\mathcal{T}V_k - V||$$

where $\mathcal{F}$ is a function space (e.g. linear, deep neural network).

Approximate value iteration (AVI)

AVI approximation error [Bertsekas & Tsitsiklis, 1996]

Let V^K be the function returned by AVI after K iterations and π_K its corresponding greedy policy. Then the performance error is bounded as

$$\|V^* - V^{\pi_K}\|_{\infty} \leq \frac{2\gamma}{(1-\gamma)^2} \max_{0 \leq k < K} \|\mathcal{T}V_k - \mathcal{AT}V_k\|_{\infty} + \frac{2\gamma^{K+1}}{1-\gamma} \|V^* - V_0\|_{\infty}$$

- If $\mathcal{A}$ is a projection in L_{∞} -norm, then $\mathcal{A}$ is a **non-expansion** and the joint operator $\mathcal{AT}$ is a contraction, which guarantees the existence of a unique fixed point $\tilde{V} = \mathcal{AT}\tilde{V}$ and thus the convergence of AVI.
- Performance error = approximation error + initialization-dependent term
- Proof (sketch):
 - Incur some (discounted) approximation error at each iteration
 - Incur some performance loss due to **partial policy evaluation** in value iteration

Approximate Q-value iteration

- Analogously to approximate value iteration:

$$Q_{k+1} = \mathcal{AT} Q_k$$

with $\mathcal{A}$ defined over $S \times A$.

- Recall: $\mathcal{T}Q(s, a) = \sum_{s'} p(s'|s, a)[r(s, a) + \gamma \max_{a'} Q(s', a')]$
 1. Computing best action from Q-values is easy
 2. Can use examples to approximate the expectation

Approximate Q-value iteration

- Unlike AVI, each iteration is amenable to solving as a **regression** problem*.
- Consider: linear approximation

$$\mathcal{F} = \left\{ \textcolor{red}{Q}_\theta(s, a) = \sum_{j=1}^d \theta_j \varphi_j(s, a), \theta \in \mathbb{R}^d \right\}$$

With features

$$\varphi_j: S \times A \rightarrow [0, L] \quad \phi(s, a) = [\varphi_1(s, a) \dots \varphi_d(s, a)]^T$$

- Each iteration, solve:

$$Q_{k+1} = \arg \min_{Q \in \mathcal{F}} \|Q - \mathcal{T} \textcolor{red}{Q}_k\|_\mu^2$$

With μ a distribution over S .

* 🖐 Pseudo-gradient method (like linear TD). **Here, it may diverge even with linear function approximation...**

Outline

- 1. Policy learning with approximate value functions**
 - a. Approximate value iteration
 - b. Fitted Q iteration**
 - c. Deep Q-network (DQN)
 - d. Double DQN (DDQN)
 - e. Rainbow DQN
 - f. Revisiting Rainbow

Fitted Q-iteration (sampling + approximate Q-iteration)

- Sample a batch of data using π , e.g., ϵ -greedy under Q_{θ_i}
 $(s_0, \mathbf{a}_0, r_0, s_1, \mathbf{a}_1, r_1, s_2, \mathbf{a}_2, r_2, \dots, s_n, \mathbf{a}_n, r_n)$

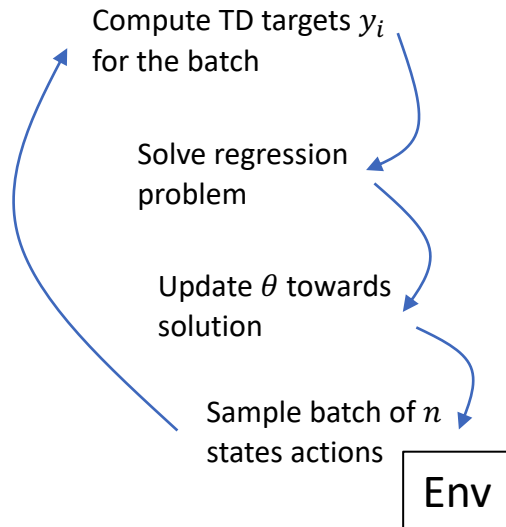
- Compute TD targets $y_t = r_t + \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a')$ for the batch

- Q-iteration **regression loss** using **bootstrapped target**

$$\begin{aligned}\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta) &= \sum_t (Q_\theta(s_t, a_t) - y_t)^2 \\ &= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a') \right)^2\end{aligned}$$

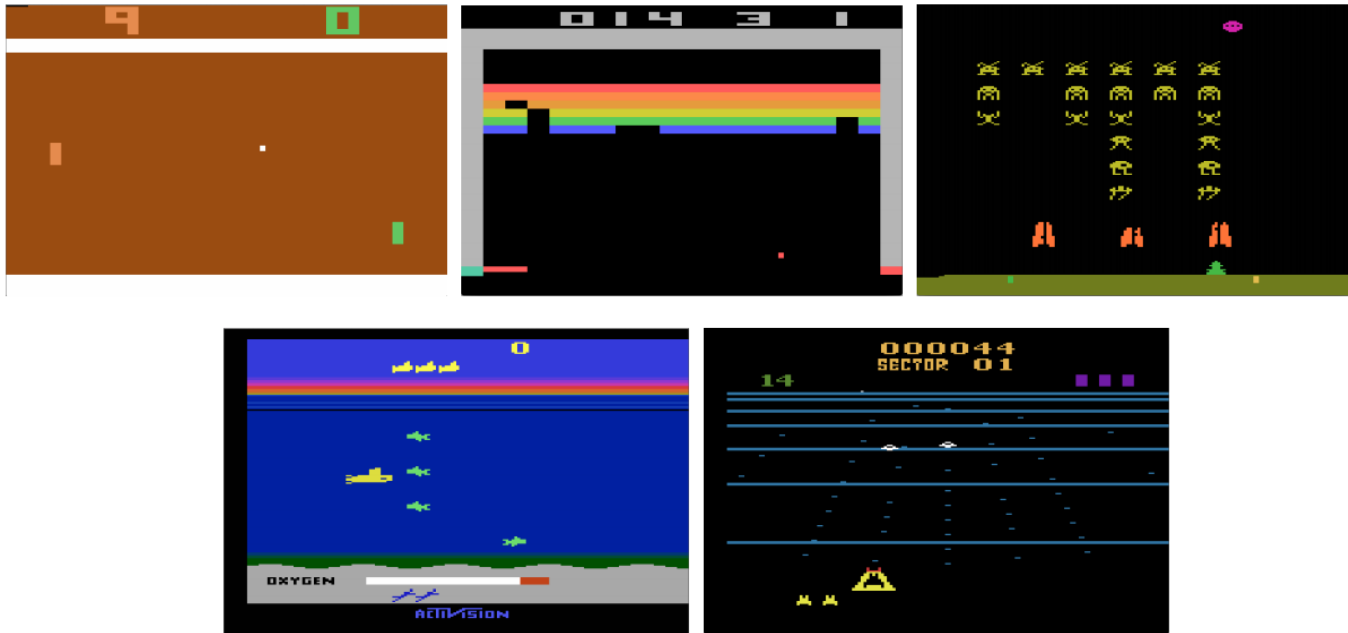
- QL **online** update with learning rate α_i

$$\begin{aligned}\hat{\theta}_{i+1} &= \hat{\theta}_i - \alpha_i \nabla_\theta \tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta) \\ &= \hat{\theta}_i - \alpha_i \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a') \right) \nabla_\theta Q_\theta(s_t, a_t)\end{aligned}$$



👉 Pseudo-gradient method. **Here, it may diverge even with linear function approximation...**

Fitted Q-iteration applied to ATARI Games



Atari

Image preprocessing: grey-scale, crop to 84x84



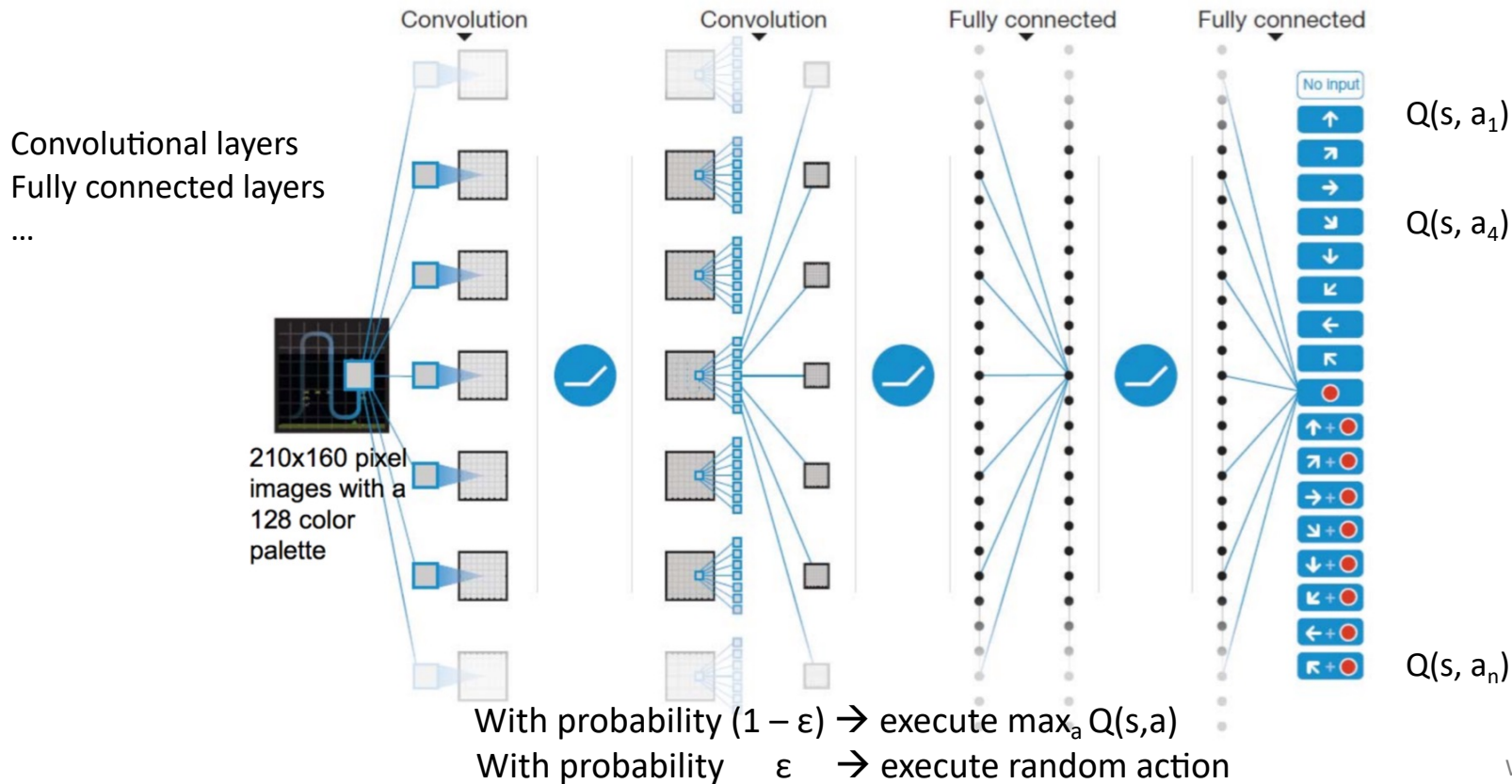
Atari

State definition: 4 last frames

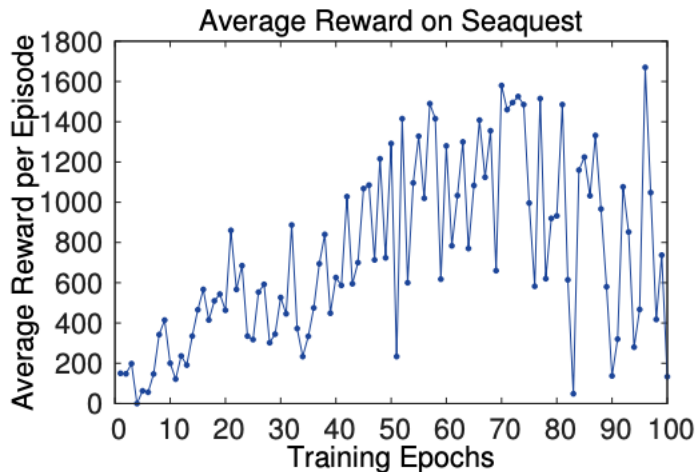
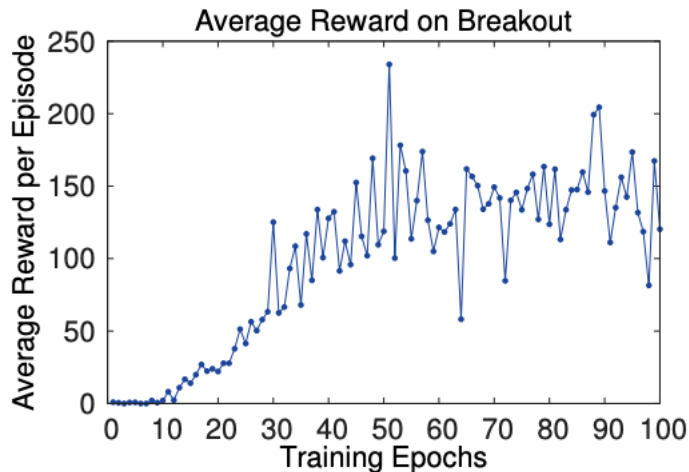
$$s_t = \left\{ \begin{array}{c} \text{Frame 1} \\ \text{Frame 2} \\ \text{Frame 3} \\ \text{Frame 4} \end{array} \right\} \left\{ \begin{array}{l} a_{t-1}, r_{t-1}, \\ \dots \\ a_{t-4}, r_{t-4} \end{array} \right\}$$


Atari

Action-value function: deepNet with as many heads as actions



Still doesn't quite work. Why?



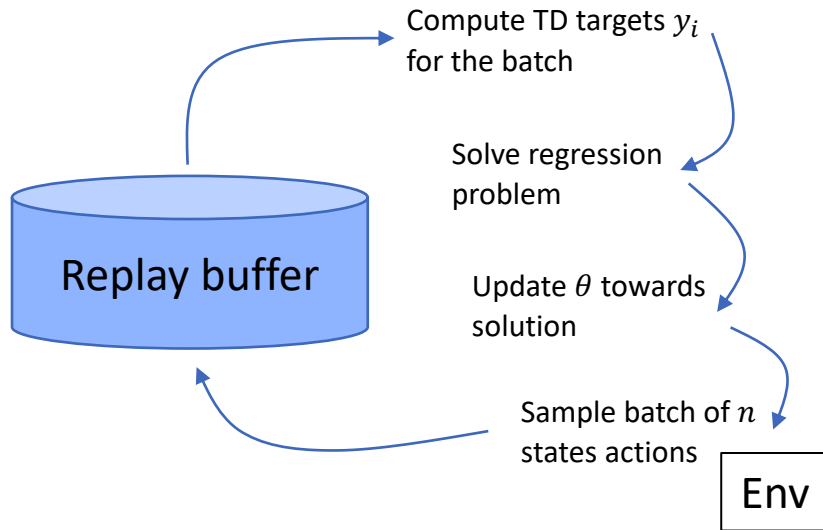
- **Recall**: Approximate QL as **Pseudo**-Gradient Descent
- **Mathematically**: Correlated samples. Violates i.i.d. assumption in supervised learning.
- **Intuition (training instability)**: Unlike in supervised learning, in RL, the agent collects its own data. If that data is bad, then the result is bad too (and may make future collection of data even worse).

Outline

1. Policy learning with approximate value functions

- a. Approximate value iteration
- b. Fitted Q iteration
- c. **Deep Q-network (DQN)**
- d. Double DQN (DDQN)
- e. Rainbow DQN
- f. Revisiting Rainbow

Solution: increase data diversity in the batch!



- De-correlates samples
- Increased diversity in data → less likely that the data overall is bad for learning

Next issue: chasing a moving target

Q-iteration loss using **bootstrapped** target

- Iteration i

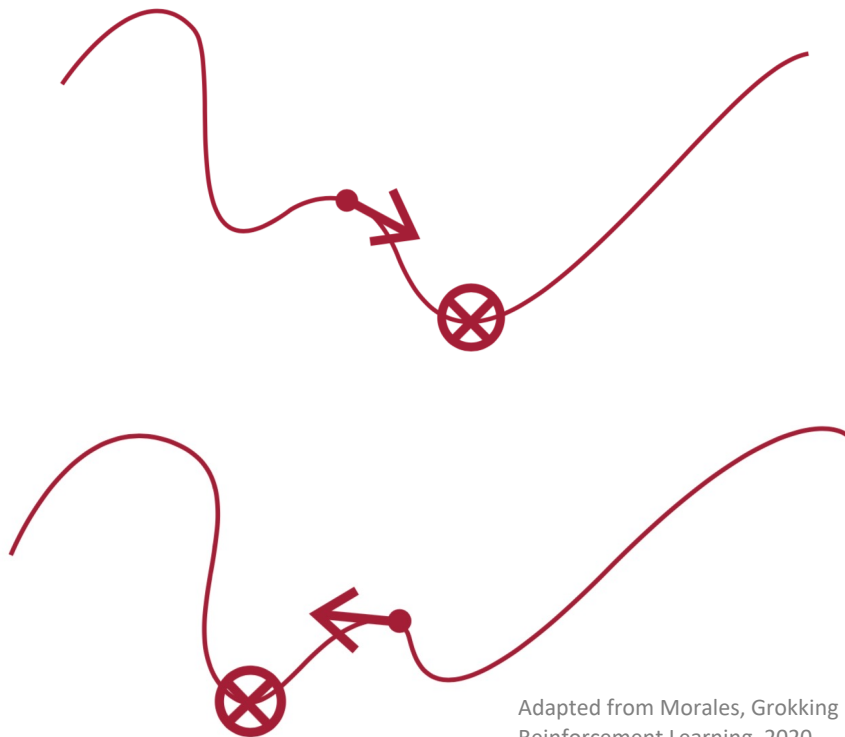
$$\begin{aligned} & \tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta) \\ &= \sum_t \left(Q_{\theta}(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a') \right)^2 \end{aligned}$$

- Iteration $i + 1$

$$\begin{aligned} & \tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta) \\ &= \sum_t \left(Q_{\theta}(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_{i+1}}(s_{t+1}, a') \right)^2 \end{aligned}$$

- Solution: change the target slowly
(e.g., $\bar{\theta} \leftarrow \theta$ every 1000 steps or $\bar{\theta}' \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$)

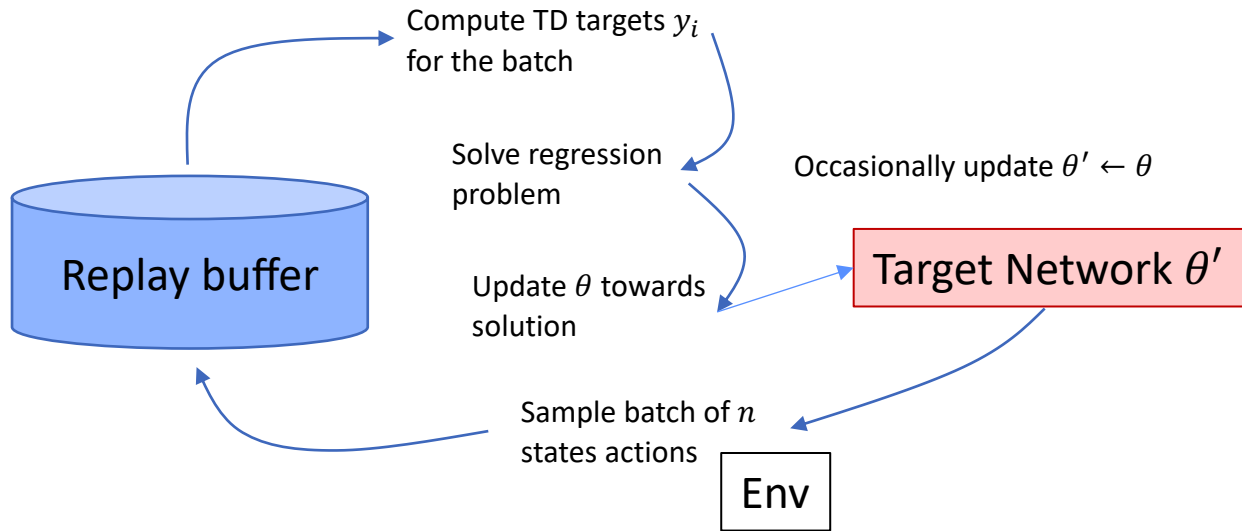
$$\begin{aligned} & \tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta) \\ &= \sum_t \left(Q_{\theta}(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\bar{\theta}}(s_{t+1}, a') \right)^2 \end{aligned}$$



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

“Target network”

Solution: slowly update a target network

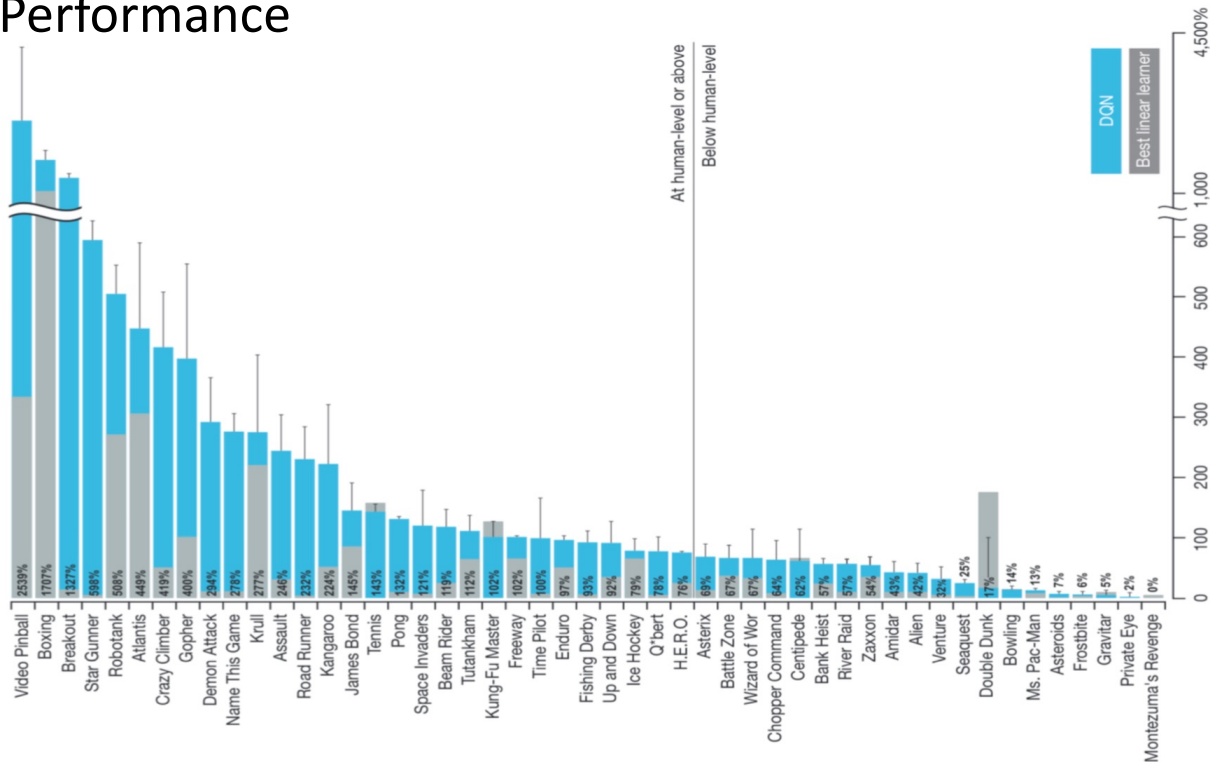


- Moves the TD target slowly $\rightarrow$ more stable training



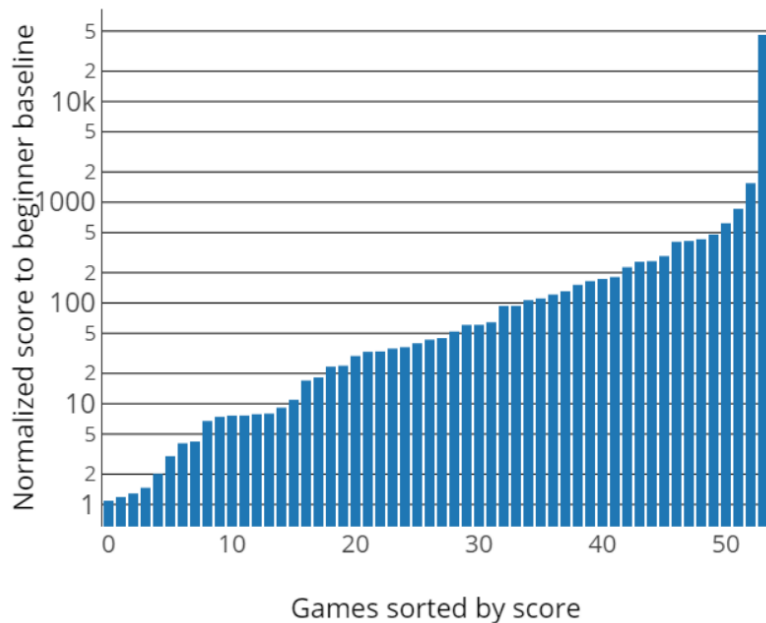
DQN – Atari

Performance



* Caveats: Human-level is indicated by 75% of the game score achieved by a professional human game tester, not necessarily an expert Atari player.

Aside: Evaluation is tricky!



* Caveats: Human-level is indicated by 75% of the game score achieved by a professional human game tester, not necessarily an expert Atari player.

Figure 2: World records scores vs. the usual beginner human baseline [16] (log scale).

DQN – Atari

Ablation

DQN

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

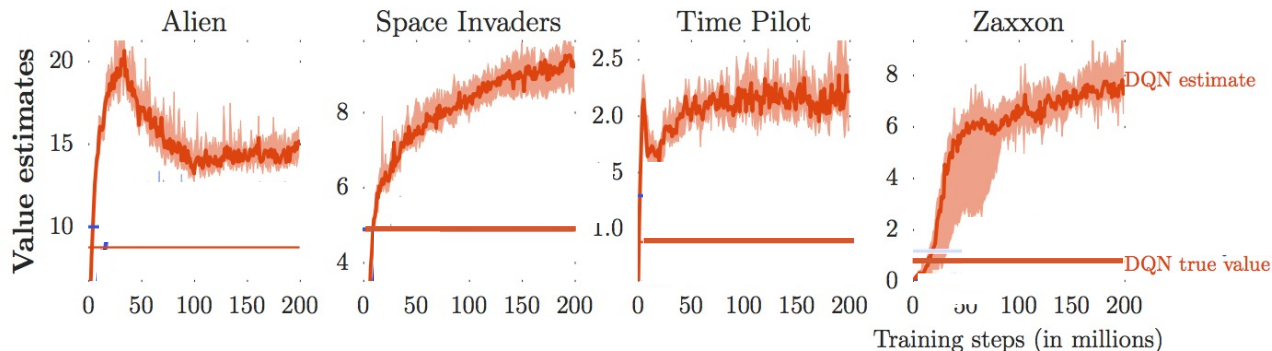
* Caveats: 1) Reported results are from a single trial. 2) Ablations were not hyperparameter tuned, while the full method was, so their performances may underestimate a similarly tuned ablation.

Outline

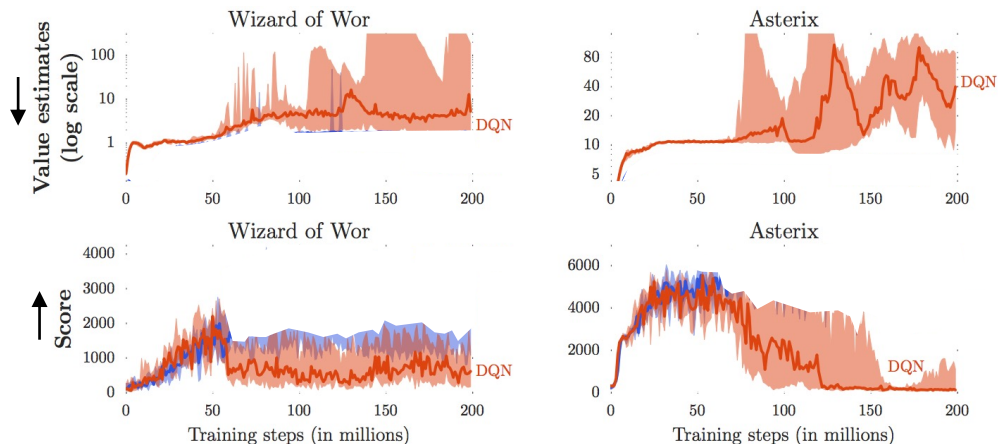
1. **Policy learning with approximate value functions**
 - a. Approximate value iteration
 - b. Fitted Q iteration
 - c. Deep Q-network (DQN)
 - d. **Double DQN (DDQN)**
 - e. Rainbow DQN
 - f. Revisiting Rainbow

Limitations of DQN

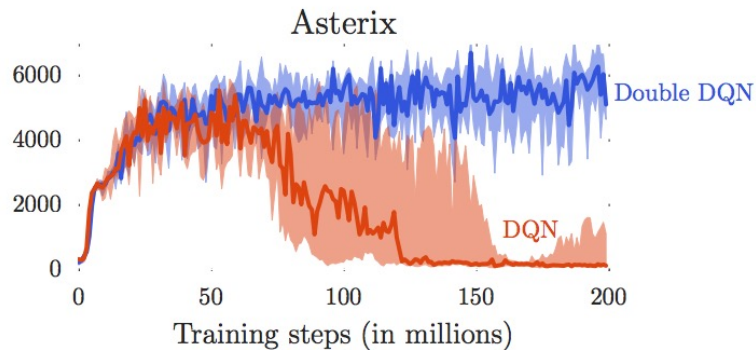
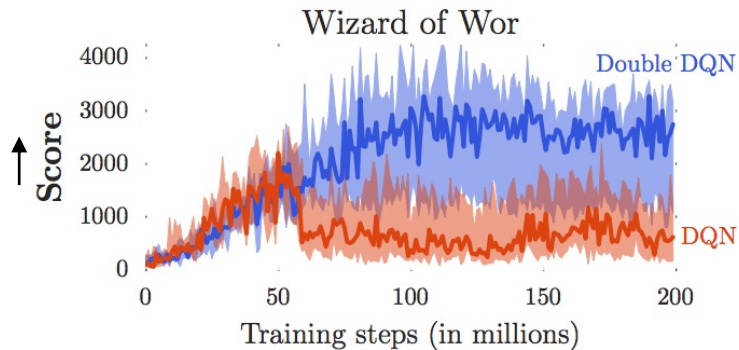
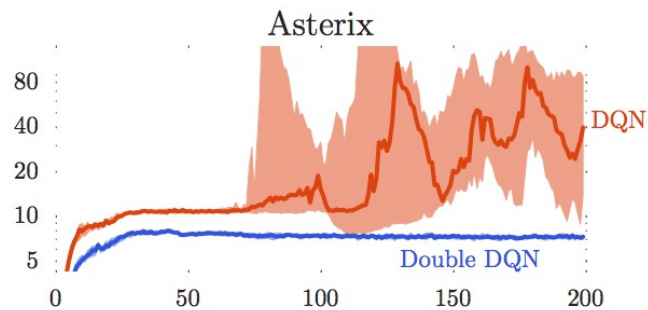
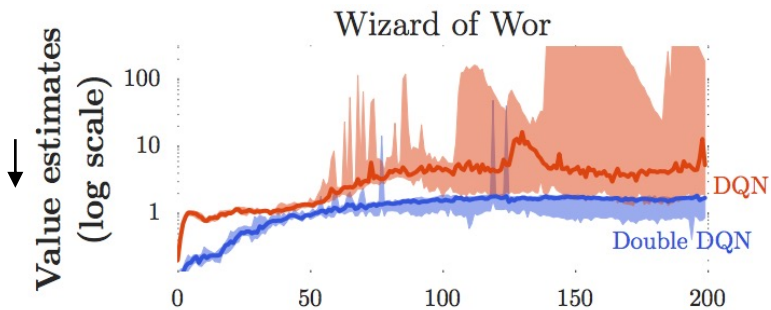
- DQN
over-estimates
Q values



- Over-estimation
leads to training
instability,
variance, &
harms overall
performance

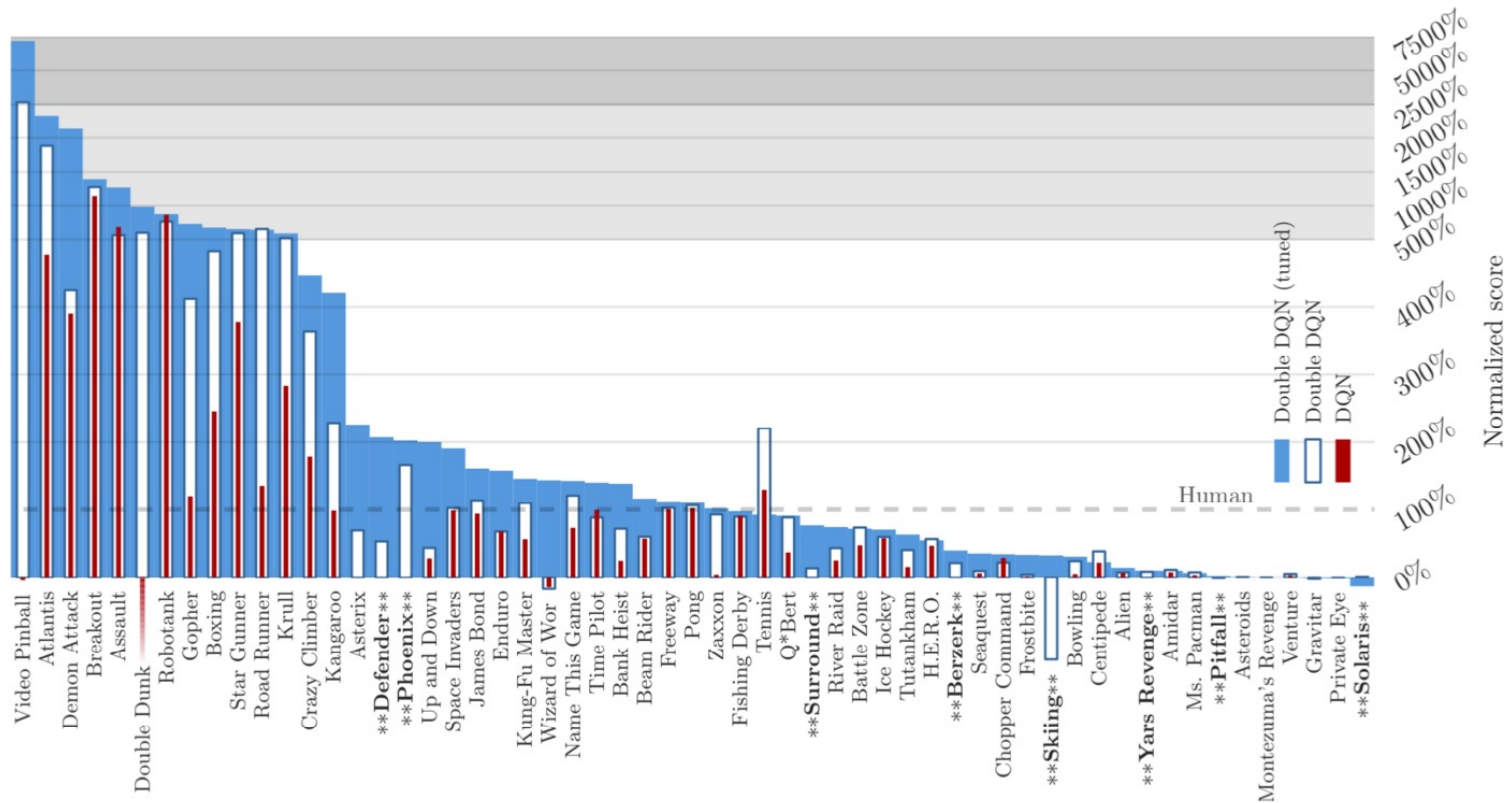


Double DQN - Atari



Double DQN - Atari

~3 lines of code
Always worth trying



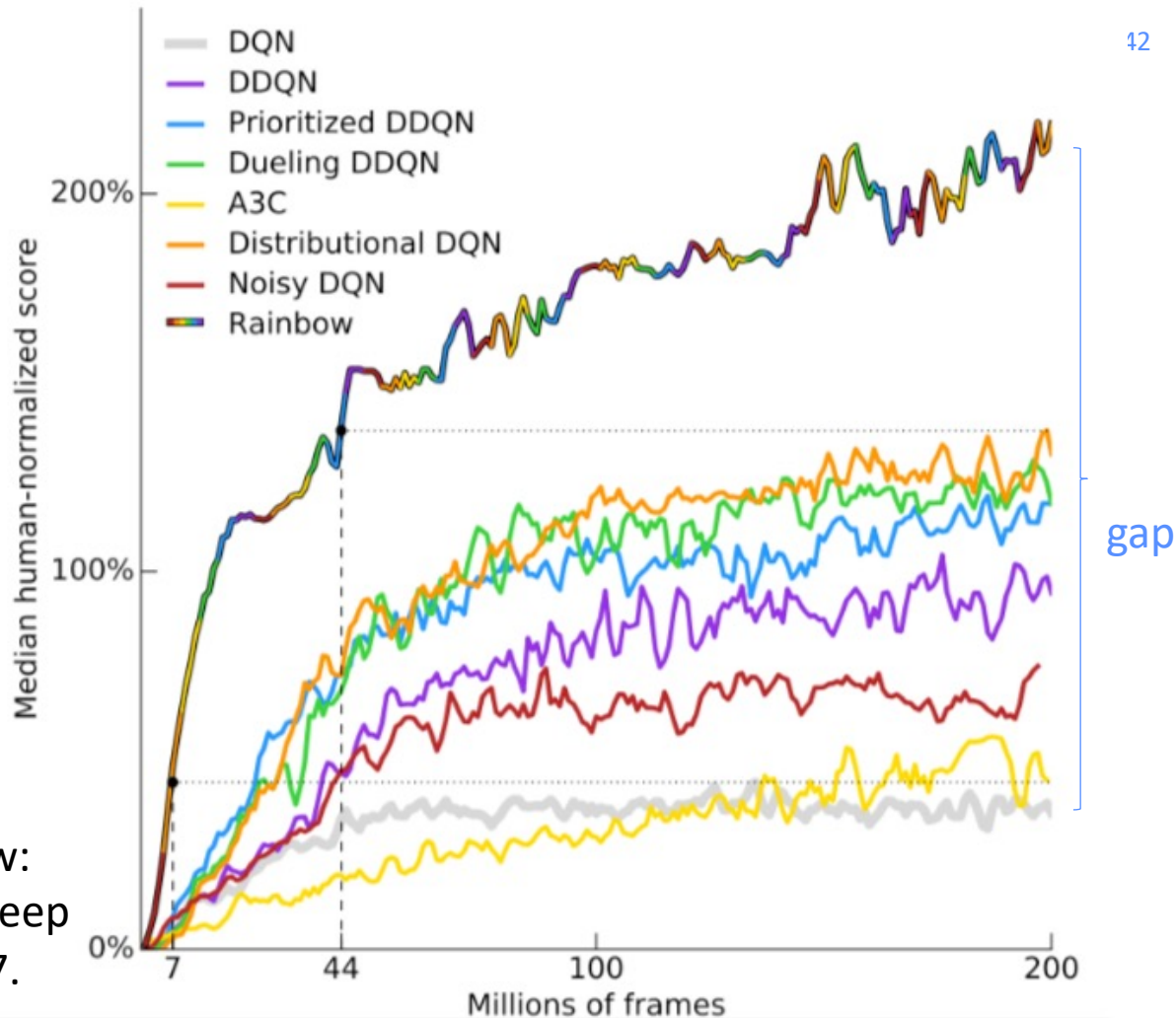
van Hasselt, Guez, Silver, 2015

Outline

1. **Policy learning with approximate value functions**
 - a. Approximate value iteration
 - b. Fitted Q iteration
 - c. Deep Q-network (DQN)
 - d. Double DQN (DDQN)
 - e. **Rainbow DQN**
 - f. Revisiting Rainbow

Rainbow DQN

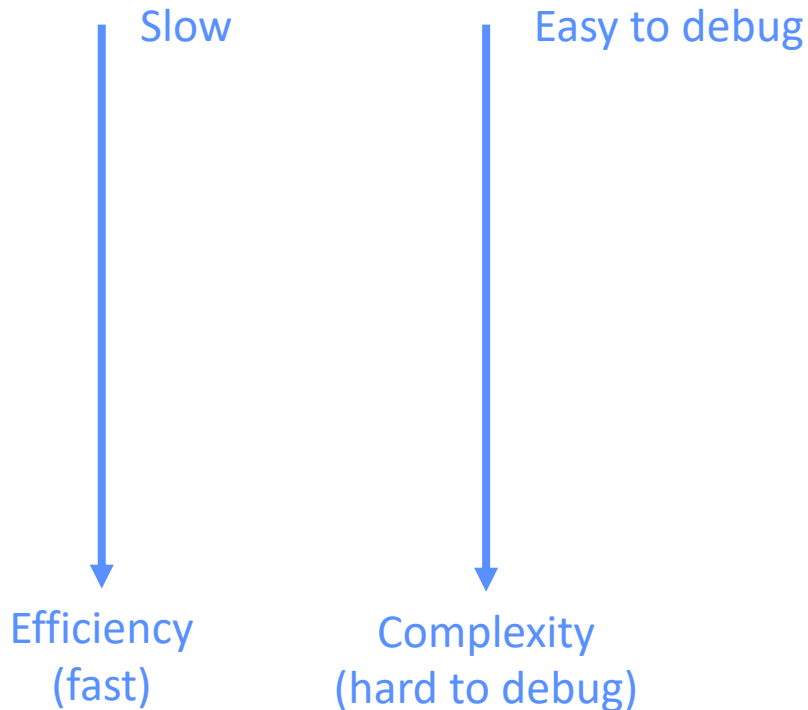
42



Hessel, Matteo, et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning." 2017.

A menu of value-based (RL) algorithms

- Dynamic programming
- Value iteration
- Q-value iteration
- Q-learning
- Fitted Q-iteration
- DQN
- DDQN
- Prioritized experience replay
- Rainbow DQN



Tip: Use the simplest algorithm the solves your problem

Outline

- 1. Policy learning with approximate value functions**
 - a. Approximate value iteration
 - b. Fitted Q iteration
 - c. Deep Q-network (DQN)
 - d. Double DQN (DDQN)
 - e. Rainbow DQN
 - f. Revisiting Rainbow**

Revisiting Rainbow: more is not always better

→ More accessible
deep RL research

Rainbow: Combining Improvements in Deep Reinforcement Learning

Matteo Hessel
DeepMind

Joseph Modayil
DeepMind

Hado van Hasselt
DeepMind

Tom Schaul
DeepMind

Georg Ostrovski
DeepMind

Will Dabney
DeepMind

Dan Horgan
DeepMind

Bilal Piot
DeepMind

Mohammad Azar
DeepMind

David Silver
DeepMind

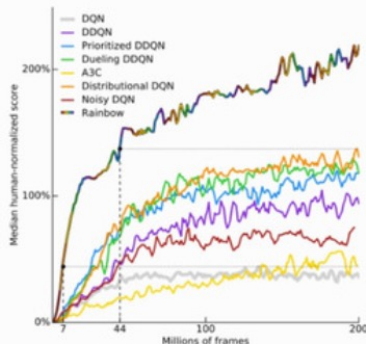


Figure 1: Median human-normalized performance across 57 Atari games. We compare our integrated agent (rainbow-colored) to DQN (grey) and six published baselines. Note that we match DQN's best performance after 7M frames, surpass any baseline within 44M frames, and reach substantially improved final performance. Curves are smoothed with a moving average over 5 points.

The cost of Rainbow:

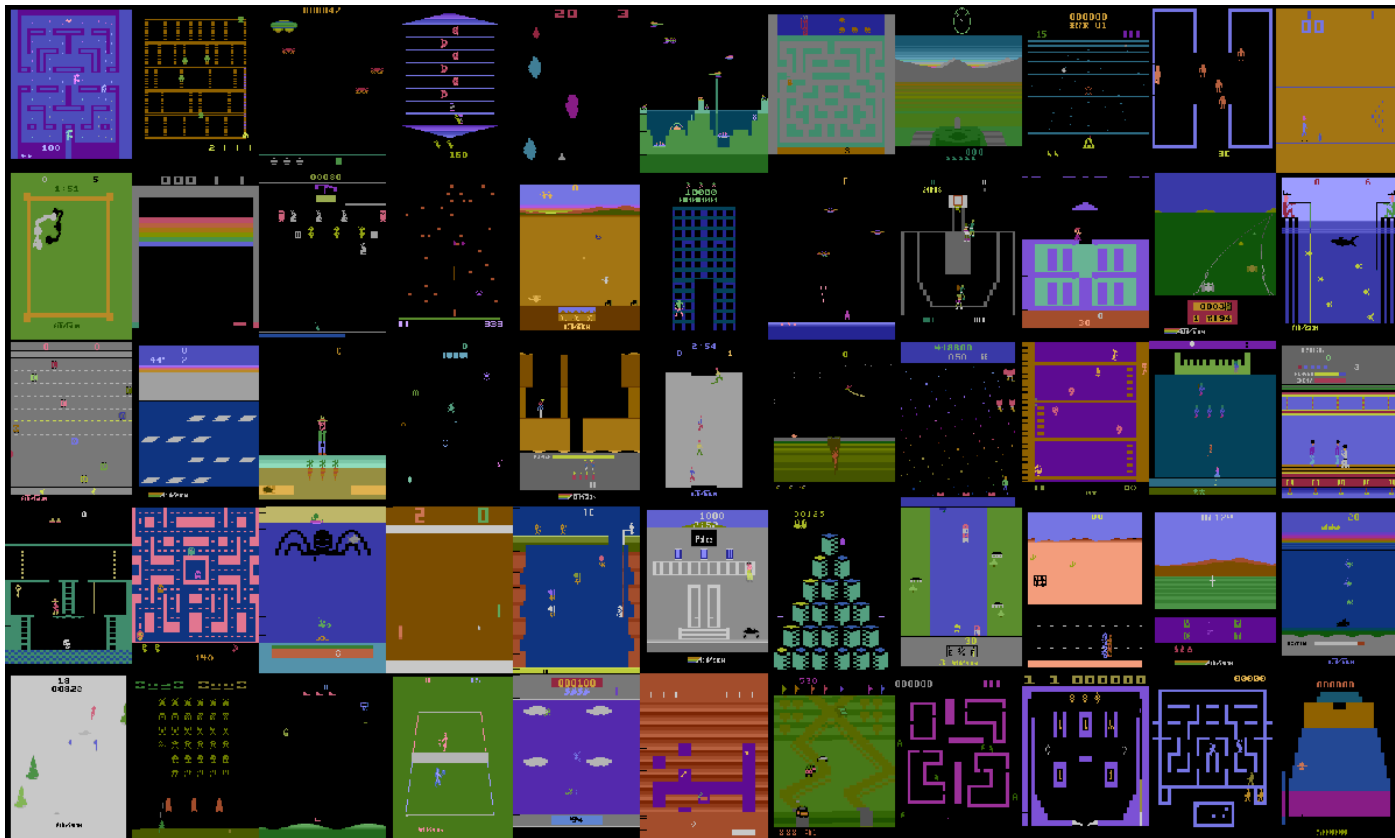
- ~5 days to train a game on a P100
- 57 games
- 5 independent seeds
- P100 costs about US\$6000

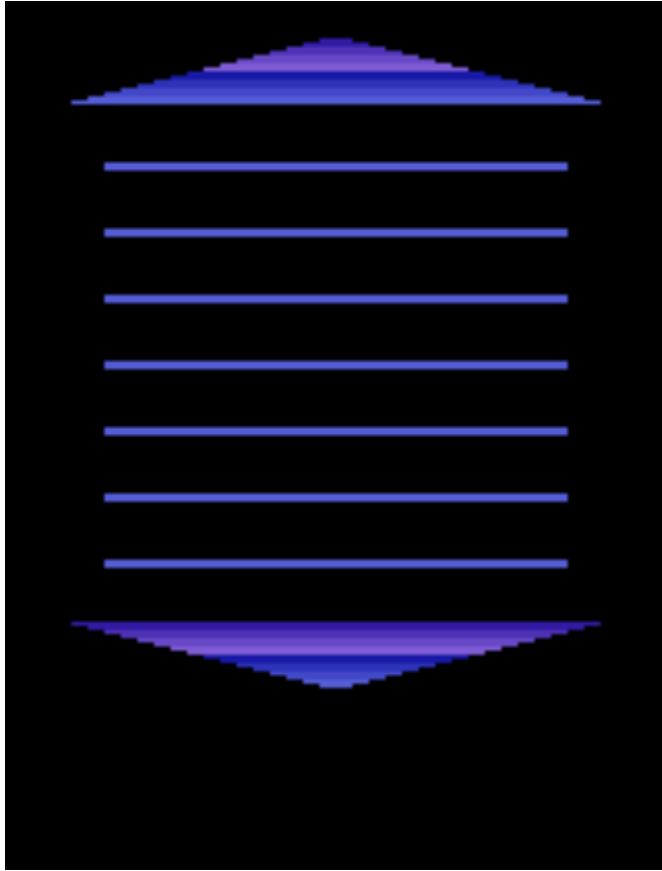
Lower-bound of cost:

- 34,200 GPU hours
- 1425 days
- Total cost: EXPENSIVE

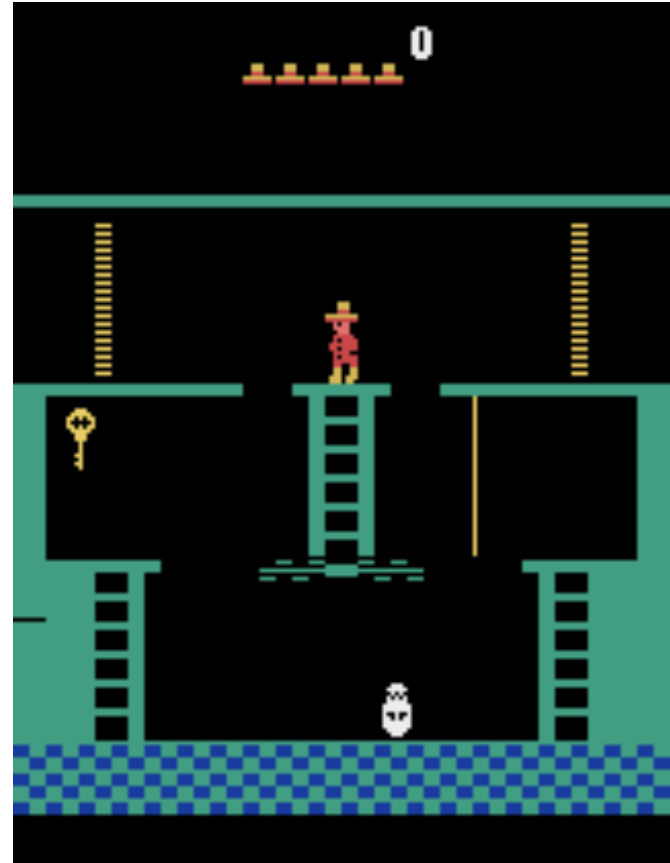
Arcade Learning Environment (ALE)

57 Atari 2600 games





Asterix



Montezuma Revenge

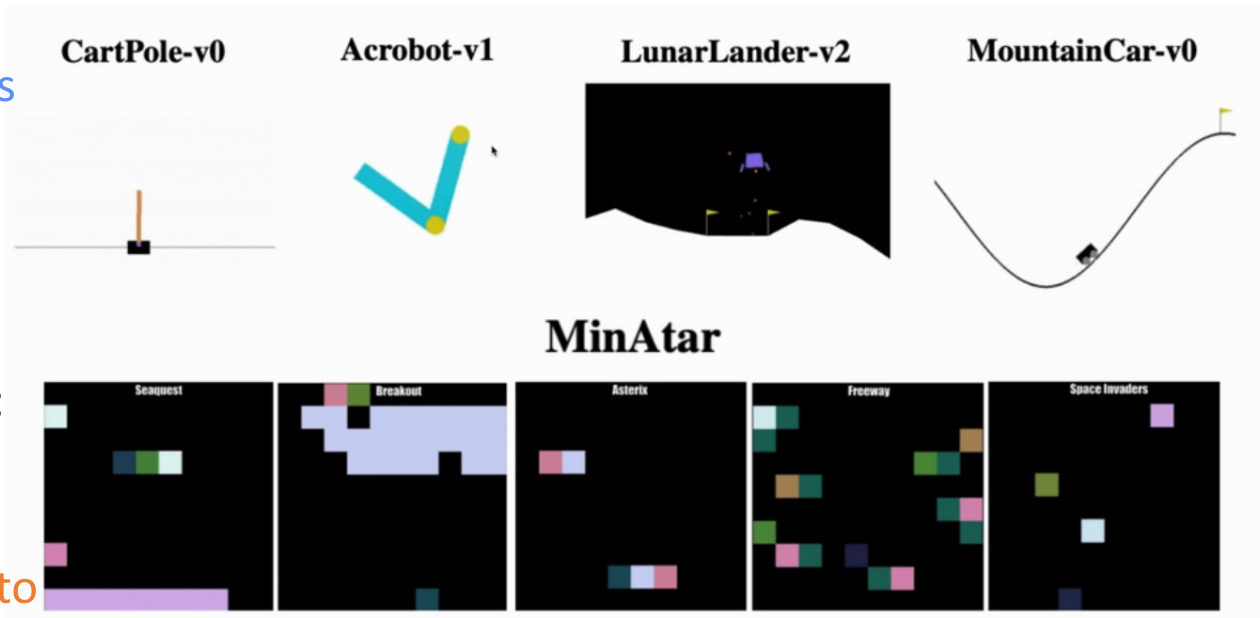
Revisiting Rainbow: more is not always better

- Careful choice of smaller yet representative tasks

- Training time for Rainbow:
34,200 GPU hours

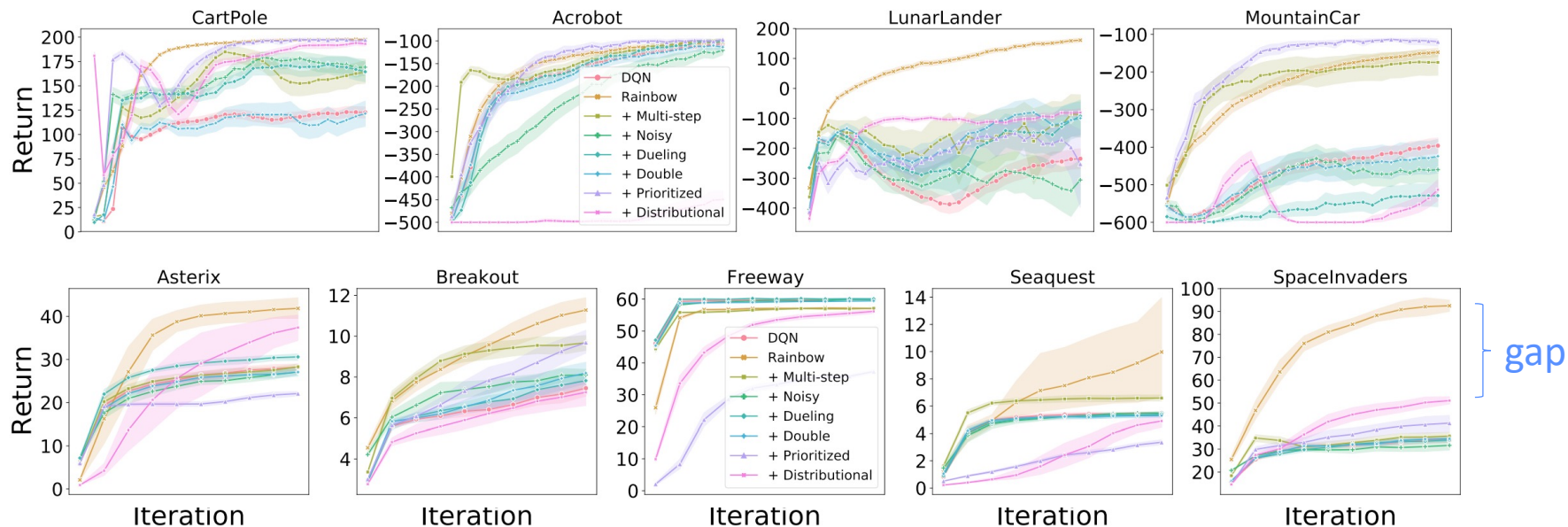
- Training time for Revisiting Rainbow:
 ≈ 78 GPU hours =
3.25 days

- 438x less compute to get the same results



Revisiting Rainbow: more is not always better

- Rainbow still $\gg$ DQN



Developments in the ALE benchmark suite

- Go-Explore (2021) tackles **hard exploration problems** (sparse rewards in Montezuma's Revenge)
- **World modeling**
 - MuZero (2020) extends AlphaGo to Atari via learned-model planning
 - Followed by DreamerV2 (2021), EfficientZero (2021), IRIS (2023), EfficientZero V2 (2024), DreamerV3 (2025)
- **Milestones**
 - Agent57 (2020) crosses the human baseline on every Atari57 game
 - Followed by MEME (2023)
 - Beyond the Rainbow (2025)
- **Meta-learning**
 - DiscoRL (2025) discover the update rule via a population of agents

How about LLMs for ALE?

- Deep RL remains the top runner for ALE
- But, ALE is a common benchmark for LLM (+RL) research
 - Transfer from Wikipedia to Atari (2022)
 - LLMs learn from Atari game manuals: Read and Reward (2023)
 - LLM inside an RL policy: LLaRP (2024)
 - LLM for RL reward design: HackAtari (2024)
 - Multi-modal LLM makes decisions from screenshots: Atari-GPT (2024)
 - LLM for summarizing trajectories and proposing actions: LLM-Explorer (2025)
 - Textual harness for Atari: TextAtari (2025)
 - LLMs for synthesizing Python code policies: Code to Control (2026)
 - ...

Summary

- **Value function approximation**: features vs function class
- Unlike its tabular counterparts, approximate value function methods may diverge, even for policy evaluation
 - **Approximate TD**, pseudo-gradient methods, and its linear version
- **Approximate Q-value iteration** shares convergence properties with **approximate value iteration** and lends itself to regression.
- **Fitted Q iteration** avoids enumeration of the state space by fitting the Q function to bootstrap targets.
 - Basis for a value-based deep RL methods, including **Deep Q Networks (DQN)**.
- **DQN**: **replay buffer**, **target networks**, **over-estimation** and other advances

References

1. *DPOC vol2 §2.5.3; [SB 16.5](#)*
2. Alessandro Lazaric. INRIA Lille. Reinforcement Learning. 2017, Lecture 4.
3. Neuro-Dynamic Programming (NDP). §3.1-3.2, Ch 6.

Reference: Detailed proof for Approximate Value Iteration

And performance loss lemma

Proof: Approximate Value Iteration

Proof. Let $\epsilon = \max_{0 \leq k < K} \|\mathcal{T}V_k - \mathcal{A}\mathcal{T}V_k\|_\infty$. This is the largest approximation error done over the iterations.

$$\begin{aligned} \|V^* - V_{k+1}\|_\infty &\leq \|\mathcal{T}V^* - \mathcal{T}V_k\|_\infty + \|\mathcal{T}V_k - V_{k+1}\|_\infty \\ &\leq \gamma \|V^* - V_k\|_\infty + \epsilon \end{aligned}$$

then

$$\begin{aligned} \|V^* - V_k\|_\infty &\leq (1 + \gamma + \dots + \gamma^{K-1})\epsilon + \gamma^K \|V^* - V_0\|_\infty \\ &\leq \frac{1}{1 - \gamma} \epsilon + \gamma^K \|V^* - V_0\|_\infty \end{aligned}$$

Since from **performance loss lemma** we have that $\|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{1-\gamma} \|V^* - V_k\|_\infty$, then we obtain

$$\|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \epsilon + \frac{2\gamma^{K+1}}{1-\gamma} \|V^* - V_0\|_\infty$$

□

From Approximation Error to Performance Loss

Proposition

Let $V \in \mathbb{R}^N$ be an approximation of V^* and π its corresponding greedy policy, then

$$\underbrace{\|V^* - V^\pi\|_\infty}_{\text{performance loss}} \leq \frac{2\gamma}{1-\gamma} \underbrace{\|V^* - V\|_\infty}_{\text{approx. error}}.$$

Furthermore, there exists $\epsilon > 0$ such that if $\|V - V^*\|_\infty \leq \epsilon$, then π is *optimal*.

Proof: Approximation Error to Performance Loss

$$\begin{aligned}\|V^* - V^\pi\|_\infty &\leq \|\mathcal{T}V^* - \mathcal{T}^\pi V\|_\infty + \|\mathcal{T}^\pi V - \mathcal{T}^\pi V^\pi\|_\infty \\ &\leq \|\mathcal{T}V^* - \mathcal{T}V\|_\infty + \gamma\|V - V^\pi\|_\infty \\ &\leq \gamma\|V^* - V\|_\infty + \gamma(\|V - V^*\|_\infty + \|V^* - V^\pi\|_\infty) \\ &\leq \frac{2\gamma}{1-\gamma}\|V^* - V\|_\infty.\end{aligned}$$

