Fall 2024

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*

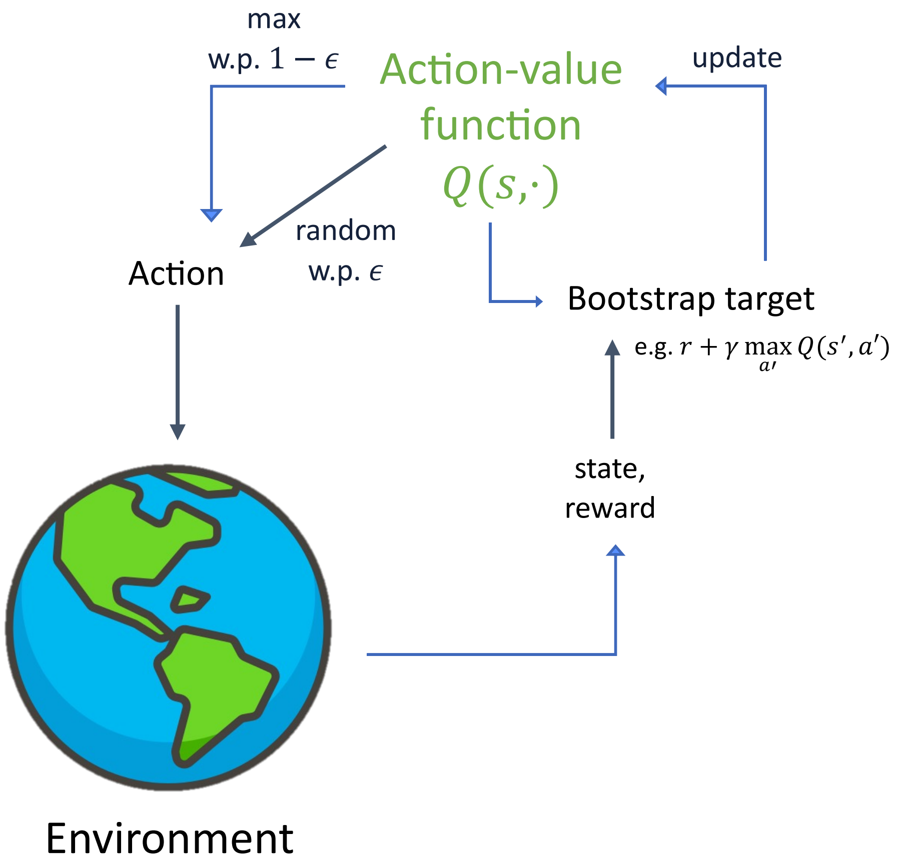
Learning milestone!

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
- …

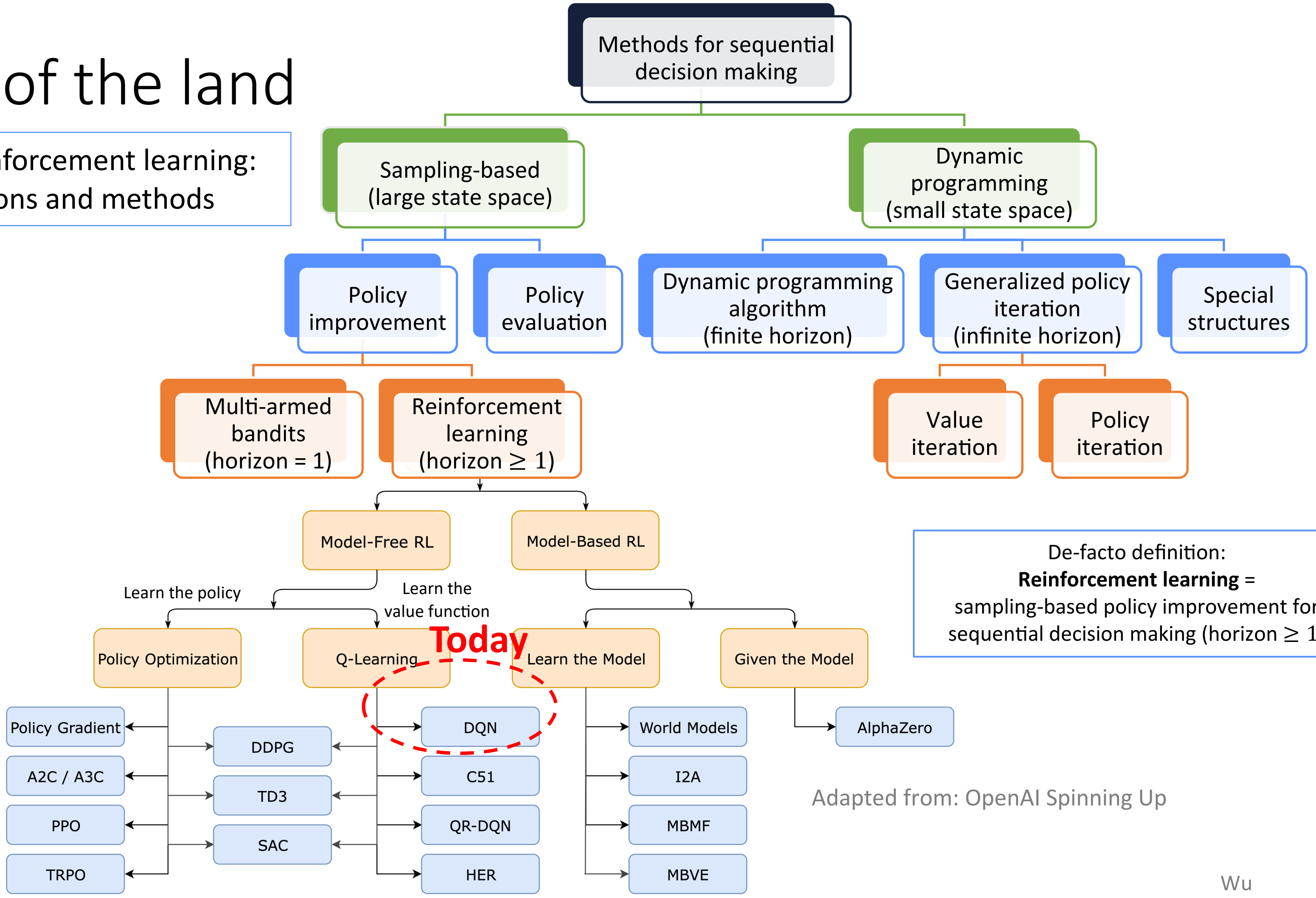


Outline

1. Approximation architectures
2. Approximate value-based algorithms

Lay of the land

6.7920: Reinforcement learning: foundations and methods



De-facto definition:
Reinforcement learning = sampling-based policy improvement for sequential decision making (horizon ≥ 1)

Adapted from: OpenAI Spinning Up

Wu

Outline

1. **Approximation architectures**
   a. Value function approximation for continuous state problems
   b. Features vs function classes
2. Approximate value-based algorithms

Notice

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

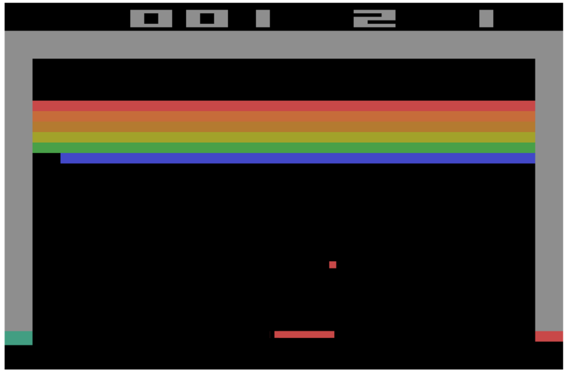
Most results smoothly extend to other settings.

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

Recall (L1): Key challenge of huge decision spaces

Arcade Learning Environment (ALE)

$a_t = \text{left}$



t

Game of Go

Possible game states: $3^{84 \times 84} \approx 10^{3366}$

Possible game states: $3^{19x19} \approx 10^{172}$

For reference:
There are between 10^{78} to 10^{82} atoms in the observable universe.

Cannot only explore. Cannot only exploit.
Must trade off exploration and exploitation.

Q-Learning: Properties

Understanding this Proposition is the main subject of today + next time.

Proposition

If the learning rate satisfies the Robbins-Monro conditions in all states $s, a \in S \times A$

$$\sum_{i=0}^{\infty} \eta_t(s,a) = \infty \qquad \sum_{i=0}^{\infty} \eta_t^2(s,a) < \infty$$

And all state-action pairs are tried infinitely often, then for all $s, a \in S \times A$

$$\hat{Q}(s,a) \xrightarrow{a.s.} Q^*(s,a)$$

- **Remark**: "infinitely often" requires a steady exploration policy.

Approximating the (state-action) value function

$Q(s,a)$

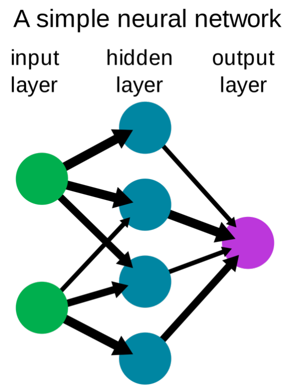
s

↑	↓	←	→
2.5	1.4	3.2	5.4
1.0	3.2	5.1	6.3
5.2	4.2	5.5	7.2
8.7	3.4	2.0	8.0
4.8	2.5	3.5	4.2
1.0	3.0	3.3	1.2
-180	-172	-99.7	-150
4.2	2.1	3.2	3.7
2.1	2.0	3.7	3.1
3.0	1.2	3.2	2.7
0.1	1.5	0.1	1.0

a

"Tabular"

$$Q_\theta(s,a) = f_\theta(s,a)$$



"Function approximation"

Value Function Approximation

Desiderata

- Expressiveness: Rich enough approximation architecture to provide close enough approximation to the function we are trying to approximate.
- Computational considerations: Effective algorithms for tuning the parameters of the approximation architecture ("training").

Example: Optimal replacement problem

State: level of wear of an object (e.g., a car).
Action: $\{(R)\text{eplace}, (K)\text{eep}\}$.
Cost:

- $c(x, R) = C$
- $c(x, K) = c(x)$ maintenance plus extra costs.

Dynamics:

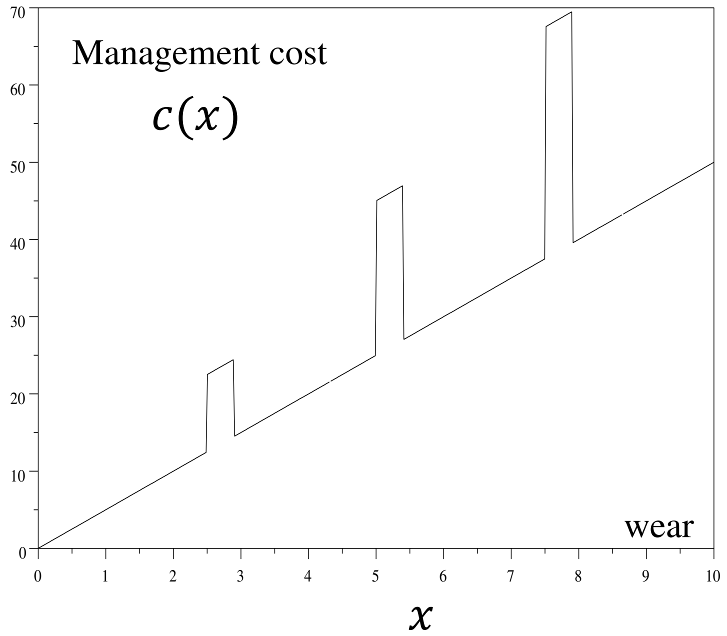
- $p(\cdot|x, R) \sim \exp(\beta)$ with density $d(y) = \beta \exp^{-\beta y} \mathbb{I}\{y \geq 0\}$,
- $p(\cdot|x, K) \sim x + \exp(\beta)$ with density $d(y - x)$.

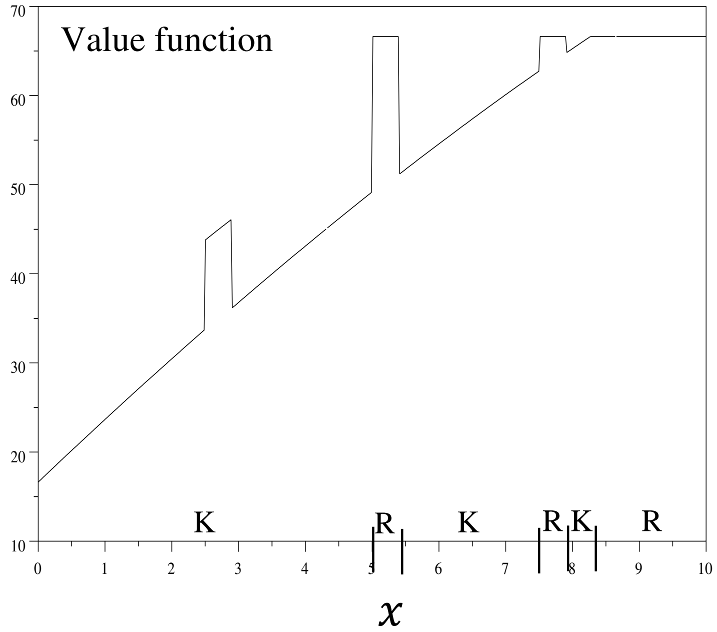
Problem: Minimize the discounted expected cost over an infinite horizon.

Optimal replacement problem

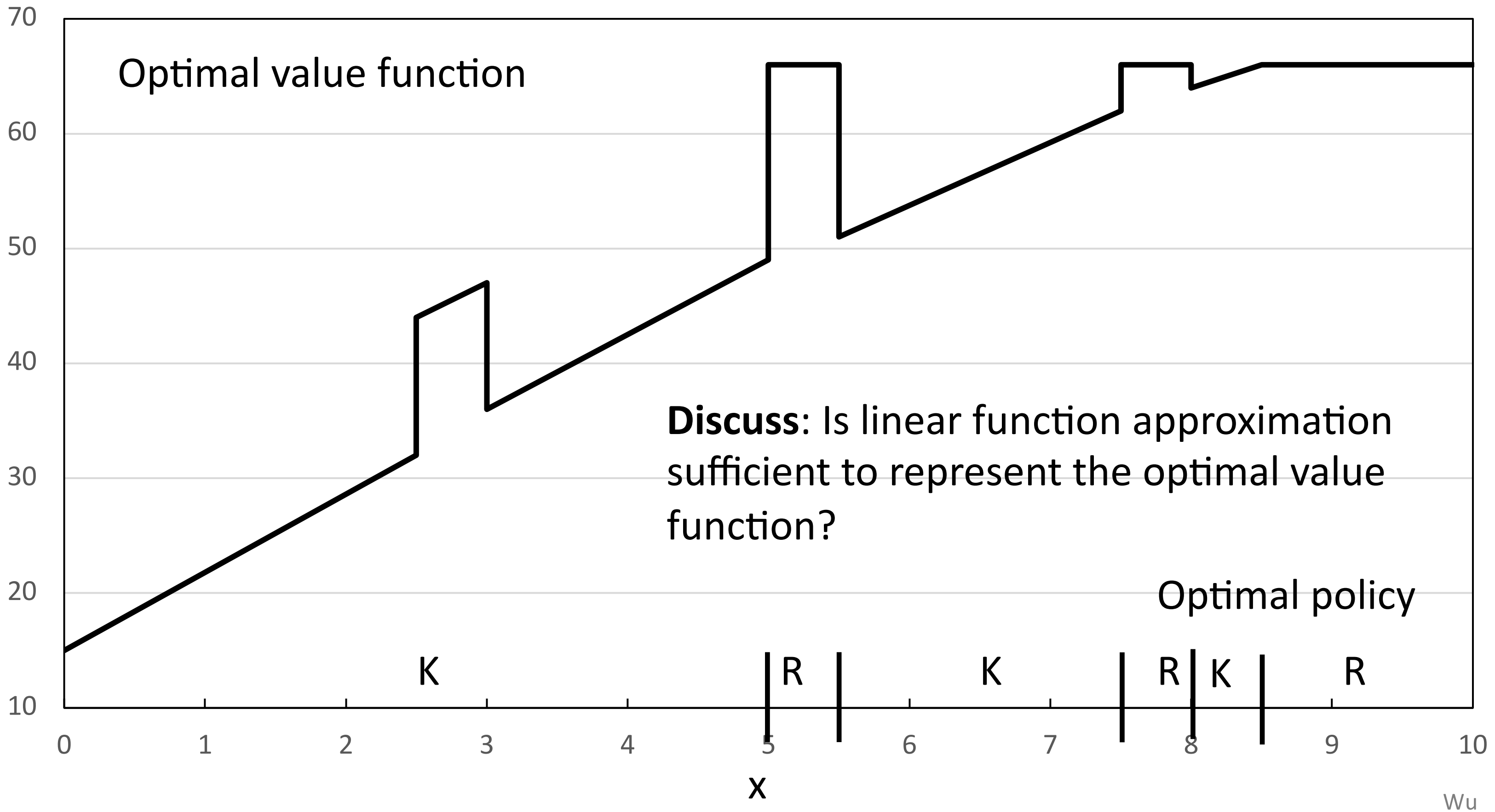
Optimal value function

$$V^*(x) = \min\left\{c(x) + \gamma \int_0^\infty d(y-x)V^*(y)dy,\ C + \gamma \int_0^\infty d(y)V^*(y)dy\right\}$$



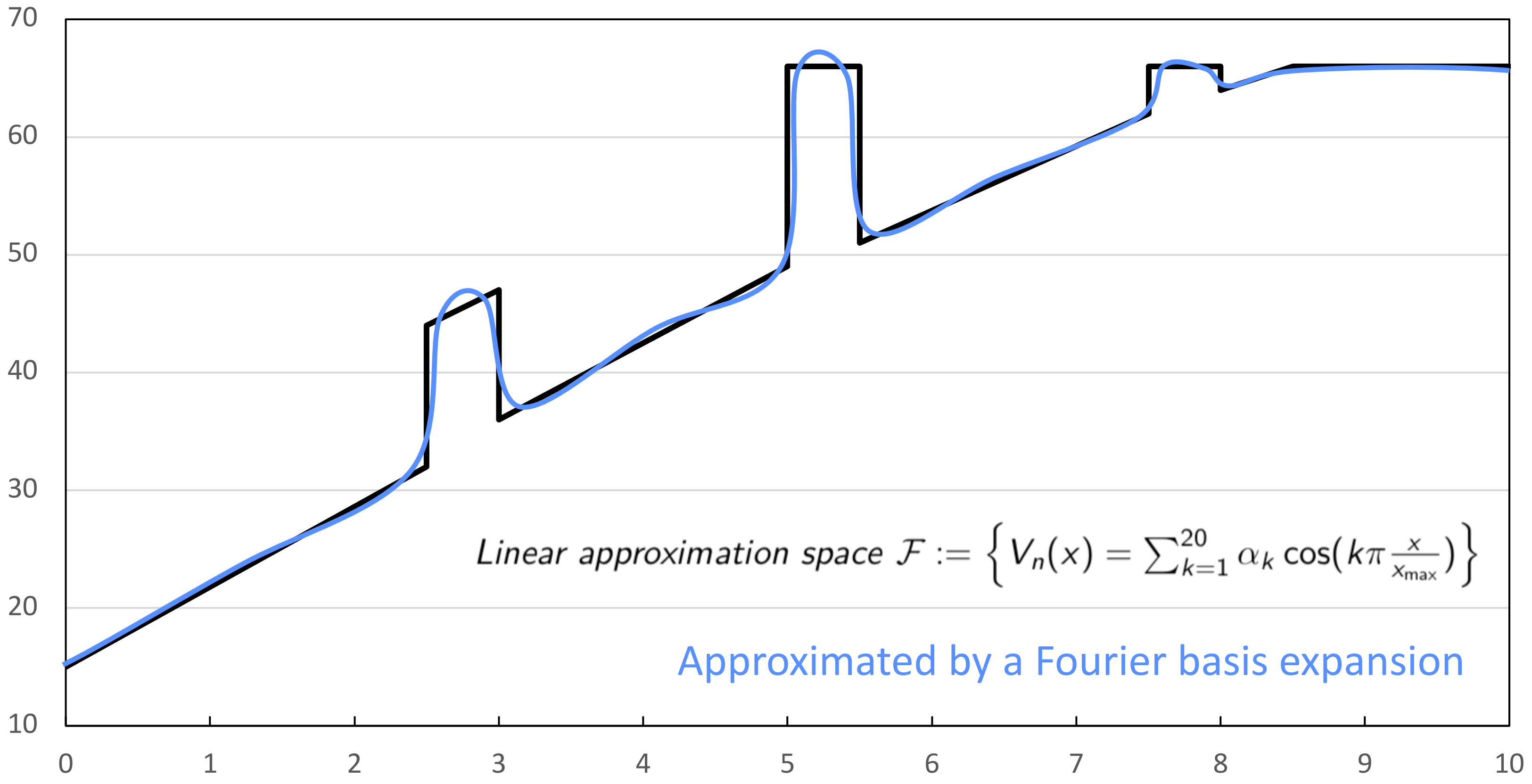


From Exact to Approximate RL



Discuss: Is linear function approximation sufficient to represent the optimal value function?

From Exact to Approximate RL



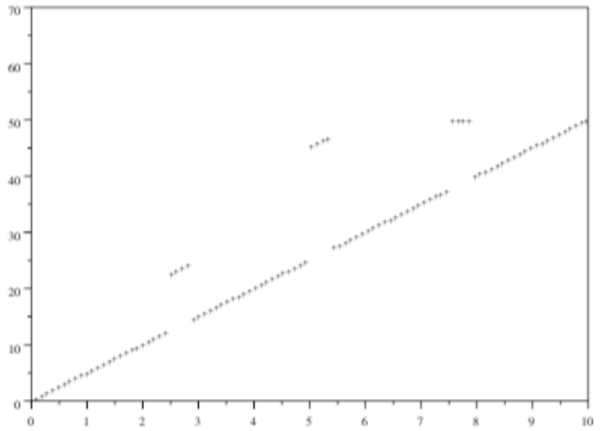
Linear approximation space $\mathcal{F} := \left\{ V_n(x) = \sum_{k=1}^{20} \alpha_k \cos\left(k\pi \frac{x}{x_{\max}}\right) \right\}$

Approximated by a Fourier basis expansion

Wu

Preview: approximate value iteration

Collect N sample on a uniform grid.



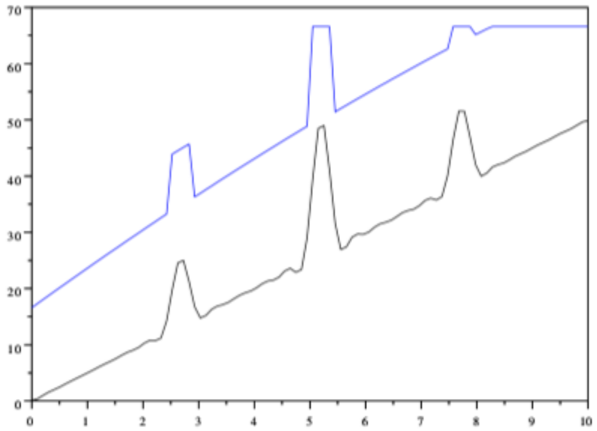


Figure: Left: the *target* values computed as $\{\mathcal{T}V_0(x_n)\}_{1\leq n\leq N}$. Right: the approximation $V_1 \in \mathcal{F}$ of the target function $\mathcal{T}V_0$.

Preview: approximate value iteration

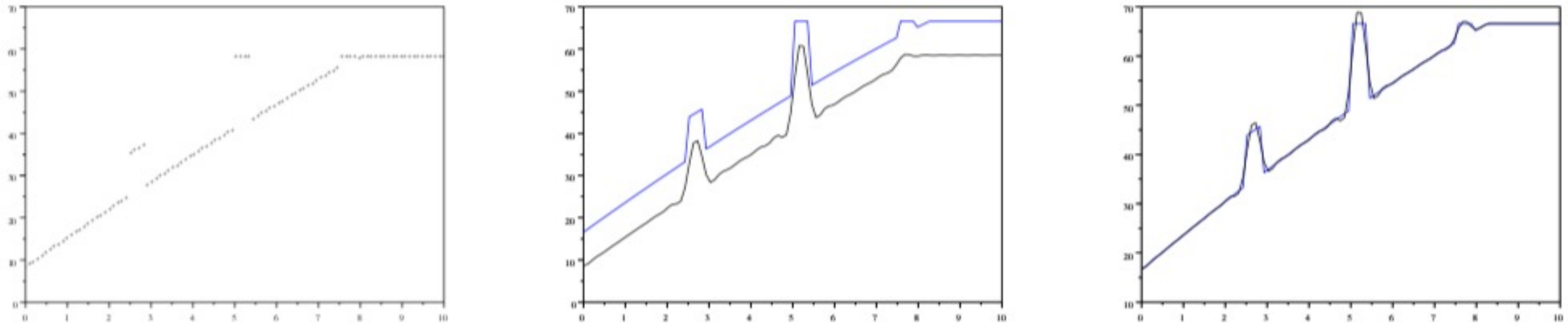


Figure: Left: the *target* values computed as $\{\mathcal{T}V_1(x_n)\}_{1\leq n\leq N}$. Center: the approximation $V_2 \in \mathcal{F}$ of $\mathcal{T}V_1$. Right: the approximation $V_n \in \mathcal{F}$ after n iterations.

Outline

1. Approximation architectures
2. **Approximate value-based algorithms**
   a. **Policy evaluation**
      - **Approximate Monte Carlo**
      - Approximate TD(0)
      - Convergence result
      - Divergence counterexample

   b. Policy learning: approximate value iteration
   c. Policy learning: fitted Q iteration
   d. Policy learning: deep Q-network (DQN)

Recall: earlier progression

- Policy evaluation
  - Monte Carlo
  - TD(1), i.e. Incremental Monte Carlo
  - TD(0)
  - TD(λ)
- Policy learning
  - SARSA
  - Q-learning

Consider: How to incorporate function approximation into these?

Policy Evaluation

Fixed policy $\boldsymbol{\pi}$

```
For $i = 1, \dots, n$
    1. Set $t = 0$
    2. Set initial state $s_0$
    3. While ($s_{t,i}$ not terminal)   [execute one trajectory]
        1. Take action $a_{t,i} = \pi(s_{t,i})$
        2. Observe next state $s_{t+1,i}$ and reward $r_{t,i} = r(s_{t,i}, a_{t,i})$
        3. Set $t = t + 1$
       EndWhile
EndFor
```

Return: Estimate of the value function $\hat{V}^{\pi}(\cdot)$

A Gentle Start: Supervised Learning

Linear space to approximate value functions

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

With features

$$\varphi_j : S \to [0, L] \qquad \phi(s) = [\varphi_1(s) \ldots \varphi_d(s)]^T$$

Approximate Monte-Carlo as Supervised Learning

- Distribution over initial states $\mathcal{D}$
- Function approximation $V_\theta : S \to \mathbb{R}, \theta \in \mathbb{R}^d$ [e.g. linear, deepNet]
- Build training set of n samples

$$s_i \sim \mathcal{D} \quad R_i = \sum_{t=0}^{H} r_{t,i} = V^\pi(s_i) + \epsilon_i \quad (\mathbb{E}[\epsilon_i] = 0)$$

Approximate Monte-Carlo as Supervised Learning

- Training (batch)

$$\hat{\theta}_n = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} L(s_i, R_i; \theta) = \frac{1}{n}\sum_{i=1}^{n}(V_\theta(s_i) - R_i)^2$$

- Testing (aka generalization error)

$$L(\hat{\theta}_n) = \mathbb{E}_{\mathcal{D}}\left[\left(V^\pi(s) - V_{\hat{\theta}_n}(s)\right)^2\right]$$

Approximate Monte-Carlo as Supervised Learning

Proposition (qualitative)

Let n be the number of samples used to build the Monte-Carlo training set. Let also $r(s,a) \in [0, r_{\max}]$ and trajectories to be as long as $H = \frac{1}{1-\gamma}$, then approximate Monte-Carlo has a generalization error:

$$L(\hat{\theta}_n) \leq \min_{\theta} L(\theta) + O\left(\frac{1}{1-\gamma}\sqrt{\frac{d}{n}}\right)$$

☞ Tends to the best possible approximation as n tends to infinity.

☟ Variance may be big.

- Proof: Apply Hoeffding's inequality

d = number of features

θ = parameterizes V_θ

Approximate Monte-Carlo as Supervised Learning

- Monte-Carlo with online training after each sample (s_i, R_i) with learning rate α_i

$$\begin{aligned}\hat{\theta}_{i+1} &= \hat{\theta}_i - \alpha_i \nabla_\theta L(s_i, R_i; \theta_i) \\ &= \hat{\theta}_i - \alpha_i \big(V_{\theta_i}(s_i) - R_i\big) \nabla_\theta V_{\theta_i}(s_i)\end{aligned}$$

Outline

1. Approximation architectures
2. **Approximate value-based algorithms**
   a. **Policy evaluation**
      - Approximate Monte Carlo
      - **Approximate TD(0)**
      - Convergence result
      - Divergence counterexample

   b. Policy learning: approximate value iteration
   c. Policy learning: fitted Q iteration
   d. Policy learning: deep Q-network (DQN)

Approximate TD(0) as Pseudo-Gradient Descent

- Run π to generate a single trajectory $(s_0, r_0, s_1, r_1, s_2, r_2, \dots, s_n, r_n)$
- TD loss using bootstrapped target

$$\tilde{L}\left(s_t, \tilde{R}_t; \theta\right) = \left(V_\theta(s_t) - \tilde{R}_t\right)^2 = \left(V_\theta(s_t) - r_t - \gamma V_{\theta_t}(s_{t+1})\right)^2$$

- TD online update with learning rate α_t

$$\begin{aligned}\hat{\theta}_{t+1} &= \hat{\theta}_t - \alpha_t \nabla_\theta \tilde{L}\left(s_t, \tilde{R}_t; \hat{\theta}_t\right) \\ &= \hat{\theta}_t - \alpha_t \left(V_{\hat{\theta}_t}(s_t) - r_t - \gamma V_{\hat{\theta}_t}(s_{t+1})\right) \nabla_\theta V_\theta\left(s_t; \hat{\theta}_t\right)\end{aligned}$$

- Not really a gradient method…
- **Discuss**: Why not?

Outline

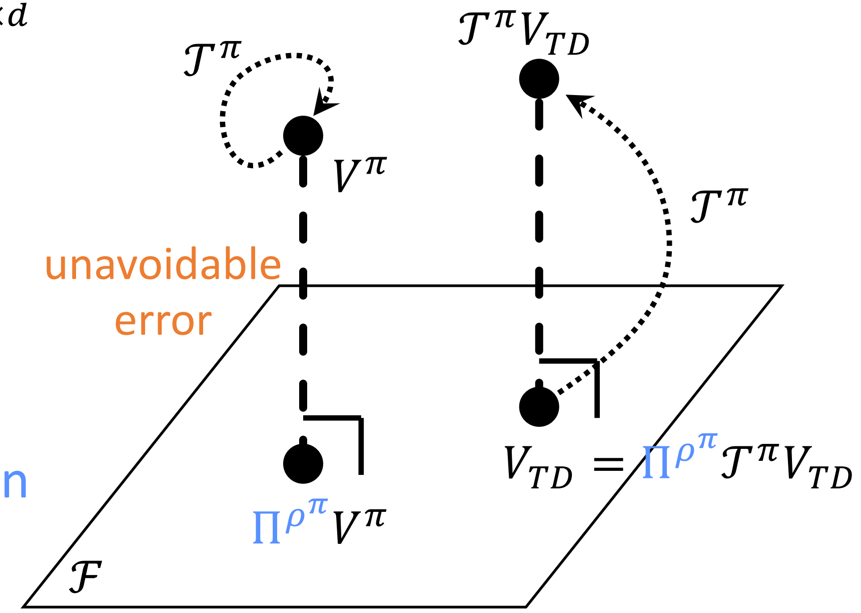
Linear TD (Least Squares TD, i.e. LSTD)

- Projection perspective (assume: stationary distribution ρ^{π})
- Compact notation: $V_{\theta} = \Phi\theta$, where
  - $\Phi = [\phi(s_1)^T; \phi(s_2)^T; \ldots \phi(s_S)^T] \in \mathbb{R}^{S\times d}$
  - $\phi(s) = [\varphi_1(s) \ldots \varphi_d(s)]^T \in \mathbb{R}^d$
- Interested in fixed point solution of

$$\Phi\theta = \Pi^{\rho^{\pi}}\mathcal{T}^{\pi}(\Phi\theta)$$

- With linear approximation, projection is linear.

$$V_{TD} = \Phi\theta^* = \Pi^{\rho^{\pi}}\mathcal{T}^{\pi}\Phi\theta^* = \Pi^{\rho^{\pi}}\mathcal{T}^{\pi}V_{TD}$$


$\mathcal{T}^{\pi}V_{TD}$
$\mathcal{T}^{\pi}$
V^{π}
$\mathcal{T}^{\pi}$
unavoidable error
$V_{TD} = \Pi^{\rho^{\pi}}\mathcal{T}^{\pi}V_{TD}$
$\Pi^{\rho^{\pi}}V^{\pi}$
$\mathcal{F}$

Linear TD (Least Squares TD, i.e. LSTD)

- Recall: $\mathcal{T}^\pi V = r + \gamma P^\pi V$

$$V_{TD} = \Phi\theta^* = \Pi^{\rho^\pi}\mathcal{T}^\pi\Phi\theta^*$$
$$= \Pi^{\rho^\pi}\mathcal{T}^\pi V_{TD}$$

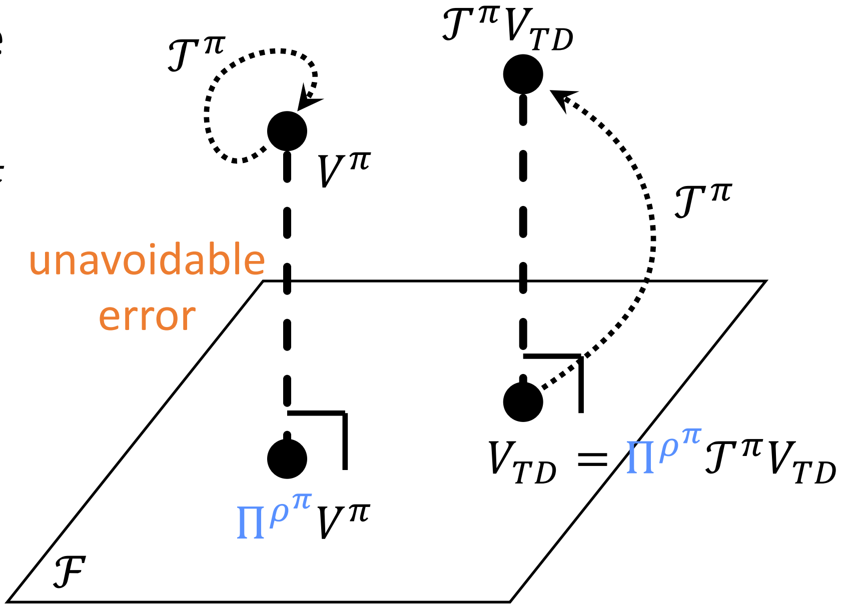
- By definition of projection (assume linearly independent features Φ), the unique solution θ^* satisfies:

$$\theta^* = \arg\min_{\theta\in\mathbb{R}^K} \|\Phi\theta - (r + \gamma P^\pi\Phi\theta)\|^2_{\rho^\pi}$$

- Setting gradient to 0, we obtain:

$$\Phi^T D_{\rho^\pi}\big(\Phi\theta^* - (r + \gamma P^\pi\Phi\theta^*)\big) = 0$$

With D_{ρ^π} a diagonal matrix with entries ρ^π.



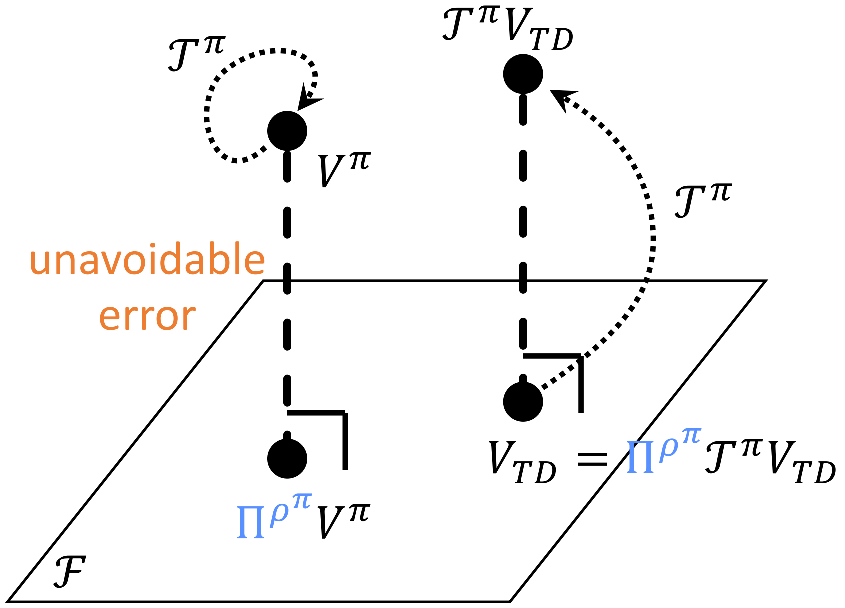
Linear TD (Least Squares TD, i.e. LSTD)

Solving for θ^*

$$\Phi^T D_{\rho^\pi}\big(\Phi\theta^* - (r + \gamma P^\pi \Phi\theta^*)\big) = 0$$

$$V_{TD} = \Phi\theta^* = \Pi^{\rho^\pi}\mathcal{T}^\pi\Phi\theta^* = \Pi^{\rho^\pi}\mathcal{T}^\pi V_{TD}$$

- Direct solution: $C\theta = d$, where $C = \Phi^T D_{\rho^\pi}(I - \gamma P^\pi)\Phi, d = \Phi^T \mathcal{D}_{\rho^\pi} r$
- Iterative method (projected VI (PVI), analogous to VI):
  $$\Phi\theta_{t+1} = \Pi^{\rho^\pi}\mathcal{T}(\Phi\theta_t)$$
  - Can write PVI explicitly: $\theta_{t+1} = \arg\min_{\theta\in\mathbb{R}^K} \|\Phi\theta - (r + \gamma P^\pi \Phi\theta_t)\|^2_{\rho^\pi}$
- Incremental variants (like TD(0), TD(λ))



Linear TD (c.f. NDP, Assumption 6.1 and Prop 6.5)

Theorem (Bradtke and Barto, 1996)

We assume that:

1. **[Stationary distribution]** There exists a distribution ρ^π over $\mathcal{S}$ such that $\lim_{t\to\infty} P^\pi(s_t = s' | s_0 = s) = \rho^\pi(s') > 0, \forall s, s' \in \mathcal{S}$; denote $\rho^\pi = \left(\rho^\pi(s_1), \rho^\pi(s_2), \dots, \rho^\pi(s_{\mathcal{S}})\right)$ and let Π^{ρ^π} be the projection with respect to the weighted Euclidean norm $\|\cdot\|_{\rho^\pi}$.
2. **[Features are full rank]** The features $(\phi_i)_{1\le k\le K\le|\mathcal{S}|}$ are linearly independent.

Then, the mappings $\mathcal{T}^{\lambda,\pi}$ and $\Pi^{\rho^\pi}\mathcal{T}^{\lambda,\pi}$ are contractions of factor $\alpha_\lambda = \frac{\gamma(1-\lambda)}{1-\gamma\lambda}$ w.r.t $\|\cdot\|_{\rho^\pi}$.

Furthermore, the linear TD estimate converges to θ^*, and $\Phi\theta^*$ is the fixed point of the projected Bellman operator:

$$\Phi\theta^* = \Pi_{\rho^\pi}\mathcal{T}^{\lambda,\pi}\Phi\theta^*$$

And it has error:

$$L_{\rho^\pi}(\theta^*) \le \frac{1}{\sqrt{1-\alpha_\lambda^2}} \min_\theta L_{\rho^\pi}(\theta)$$

Where L_{ρ^π} is the expected loss w.r.t. the stationary distribution ρ^π.

☞ Linear TD converges.

☞ Error is related to the best possible error.

☞ $\gamma = 0 \to$ accurate. **Discuss: Why?**

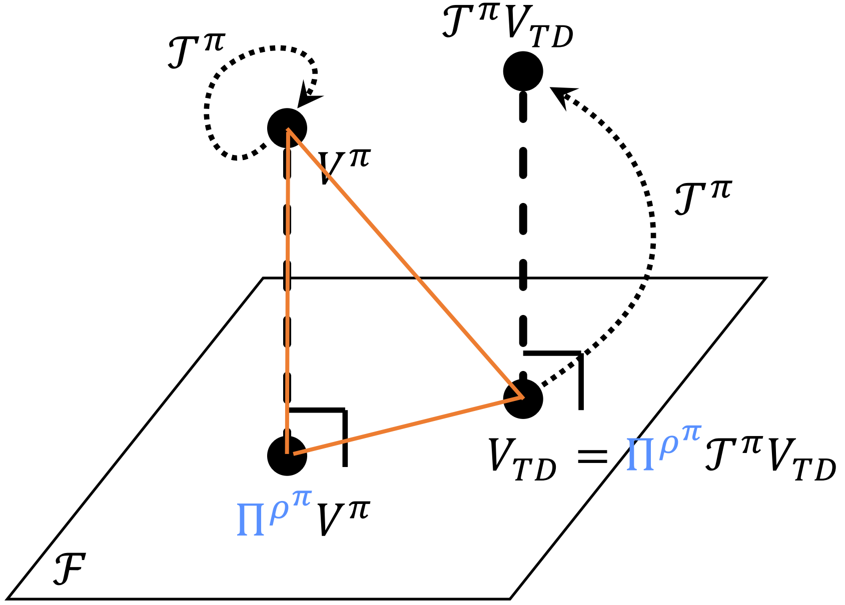
☞ $\lambda = 0 \to$ inaccurate. **Discuss: Why?**

Proof (sketch): Linear TD ($\lambda = 0 \rightarrow \alpha_0 = \gamma$)

1. Show that $\Pi^{\rho^\pi}\mathcal{T}^\pi$ is a contraction in L_{2,ρ_π} with a unique fixed point V_{TD}.
   - $\mathcal{T}^\pi$ is a contraction
   - Π^{ρ^π} is a non-expansion

2. Bound the error using the Pythagorean theorem.

$$V_{TD} = \Phi\theta^* = \Pi^{\rho^\pi}\mathcal{T}^\pi\Phi\theta^*$$
$$= \Pi^{\rho^\pi}\mathcal{T}^\pi V_{TD}$$



Approximate TD

Approximate TD may not converge (i.e. it might diverge) if:

- Linear approximation but states s_i are obtained by following a different policy (off-policy learning)
- Non-linear approximation and states s_i are obtained by following π

Outline

1. Approximation architectures

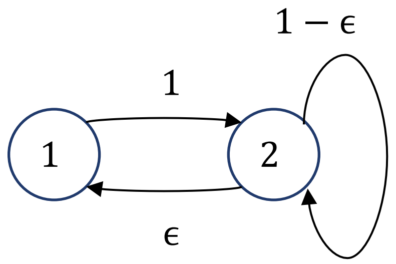
2. **Approximate value-based algorithms**
   a. **Policy evaluation**
      - Approximate Monte Carlo
      - Approximate TD(0)
      - Convergence result
      - **Divergence counterexample**

   b. Policy learning: approximate value iteration
   c. Policy learning: fitted Q iteration
   d. Policy learning: deep Q-network (DQN)

Counterexample: Divergence of Off-policy Linear TD(0)

- Consider the Markov chain induced by π, with two states $S = \{1, 2\}$. Reward is always 0.
- Consider a $\rho' \neq \rho^{\pi}$, that chooses the next state {1,2} with equal probability.
- Linear function approximation with parameter $w \in \mathbb{R}$
  - $V_w(s) = w \cdot s$
  - Note: $\phi(s) \coloneqq s$, that is, $\phi(1) = 1, \phi(2) = 2$
- Recall linear TD update:

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \alpha_t(\phi_t^T \theta_t - r_t \ - \gamma \phi_{t+1}^T \theta_t)\phi_t$$



Compare

- TD(0) with states sampled from ρ^{π} (converges)

$$\mathbb{E}[w_{t+1}] = \mathbb{E}[w_t] - \alpha_t 4(1-\gamma)\mathbb{E}[w_t] + O(\epsilon)\mathbb{E}[w_t]$$

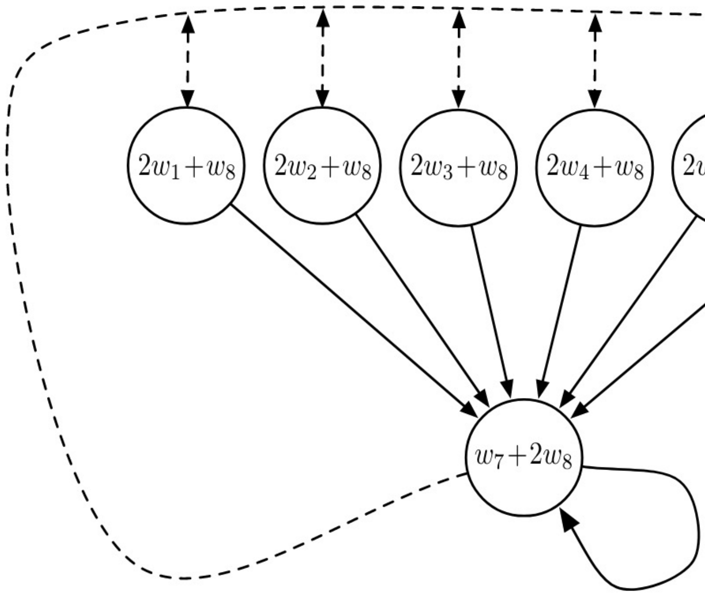
- TD(0) with states sampled from ρ' (diverges for $\gamma > 3/4$, ϵ small enough, if $w_0 \neq 0$)

$$\mathbb{E}[w_{t+1}] = \mathbb{E}[w_t] - \alpha_t \frac{1}{2}(1-2\gamma)\mathbb{E}[w_t] - \alpha_t \frac{1}{2}\big((1-\epsilon)2(1-\gamma) + \epsilon(2-\gamma)\big)\mathbb{E}[w_t]$$

$$= \ \mathbb{E}[w_t] - \alpha_t \frac{1}{2}(3-4\gamma)\mathbb{E}[w_t] + O(\epsilon)\mathbb{E}[w_t]$$

Further reading: NDP §6.3

Numerical example: Baird's counterexample



Similar counterexamples for Q-learning

Weights $w \equiv \theta$

$\pi(\text{solid}|\cdot) = 1$ Target policy

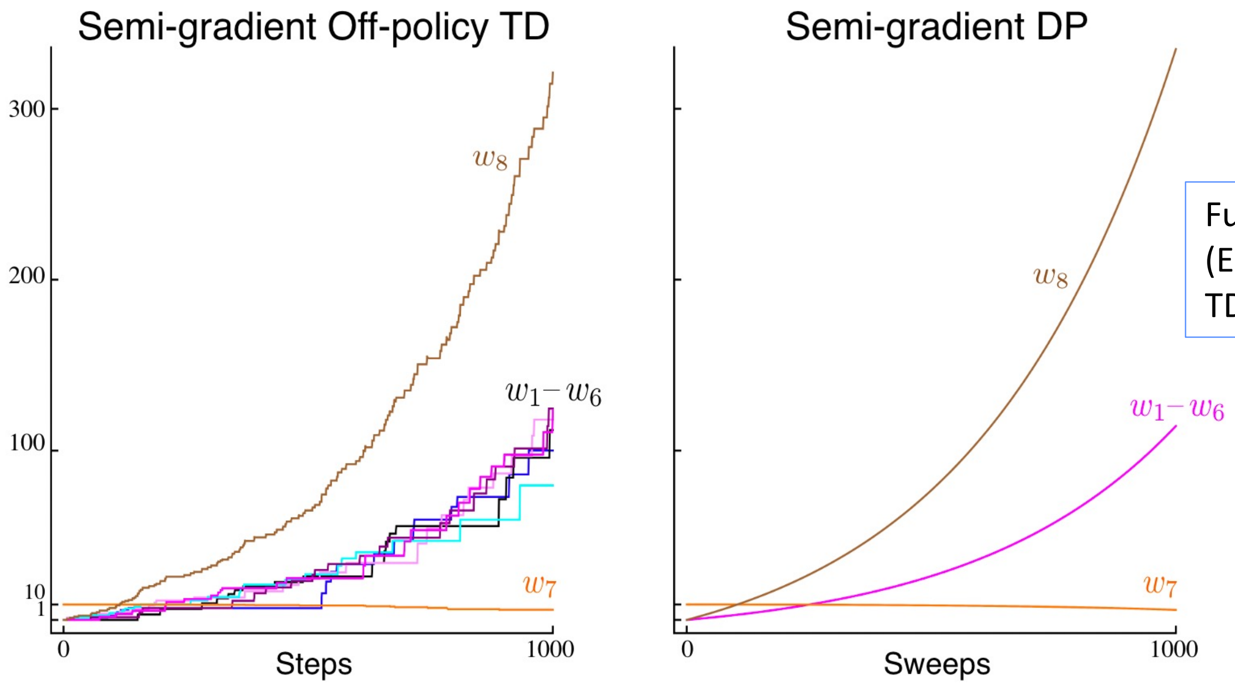
$b(\text{dashed}|\cdot) = 6/7$
$b(\text{solid}|\cdot) = 1/7$
Behavior policy

$\gamma = 0.99$

Figure 11.1: Baird's counterexample. The approximate state-value function for this Markov process is of the form shown by the linear expressions inside each state. The solid action usually results in the seventh state, and the dashed action usually results in one of the other six states, each with equal probability. The reward is always zero.

Further reading: S&B §11.2.

Numerical example: Baird's counterexample



Further reading: See NDP §6.3 (Example 6.6) for a nonlinear TD(0) counterexample

Figure 11.2: Demonstration of instability on Baird's counterexample. Shown are the evolution of the components of the parameter vector $\mathbf{w}$ of the two semi-gradient algorithms. The step size was $\alpha = 0.01$, and the initial weights were $\mathbf{w} = (1, 1, 1, 1, 1, 1, 10, 1)^\top$.

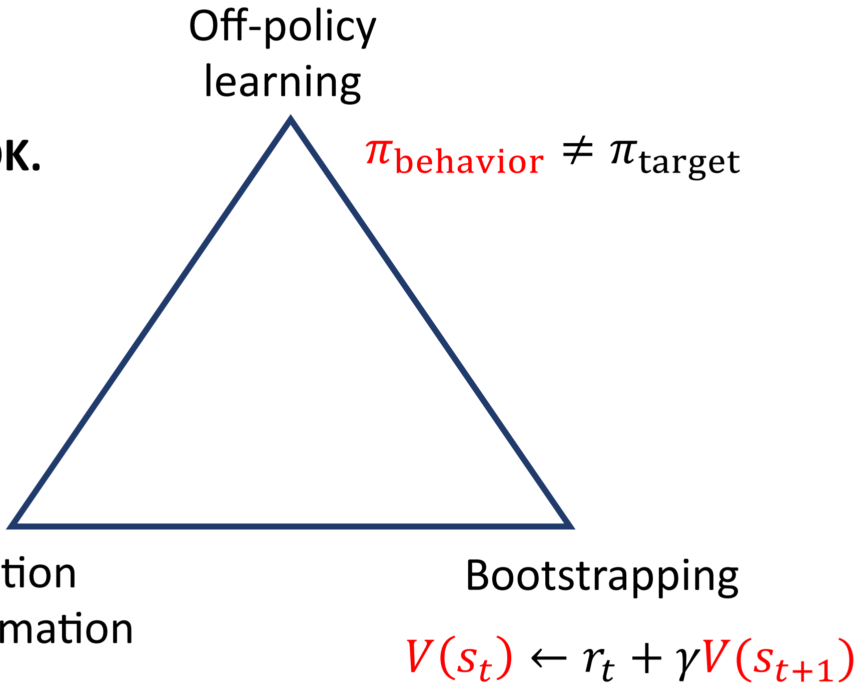
Weights $w \equiv \theta$

Further reading: S&B §11.2.

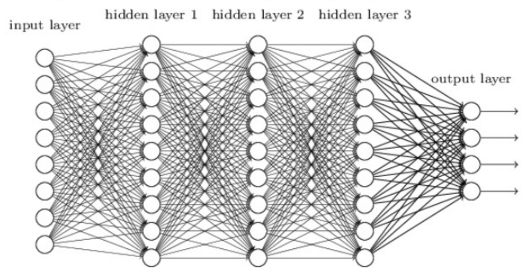
The deadly triad

The risk of divergence arises whenever we combine:

Claim: Any two without the third is OK.


Off-policy learning
$\pi_{\text{behavior}} \neq \pi_{\text{target}}$
Function approximation
Bootstrapping
$V(s_t) \leftarrow r_t + \gamma V(s_{t+1})$

$Q(s,a) \approx f_\theta(s,a)$



Further reading: S&B §11.3

Possible remedies (and further readings)

- More careful algorithm designs [Sutton et al., 2009; S&B §11.7-11.8]
  - “Fast gradient-descent methods for temporal-difference learning with linear function approximation.” ICML.
    - Gradient TD (GTD)
    - TD with gradient correction (TDC)
  - Emphatic TD [Sutton et al., 2016], etc.
    - “An emphatic approach to the problem of off-policy temporal-difference learning.” JMLR.
  - Convergence guarantees for off-policy and “mildly” non-linear approximators
- Reducing variance [S&B §11.9]
- Using a target network [Mnih et al., 2015, Zhang et al., 2021]
  - “Human-level control through deep reinforcement learning.” Nature.
  - “Breaking the deadly triad with a target network.” ICML.
  - Target network Q_{target}: periodically synced by the value network
  - Value network Q: updated via gradient methods
  - Key ingredients in (double) deep Q-learning (DQN).

Outline

1. Approximation architectures

2. **Approximate value-based algorithms**
   a. Policy evaluation
   b. **Policy learning: approximate value iteration**
   c. Policy learning: fitted Q iteration
   d. Policy learning: deep Q-network (DQN)

Approximate value iteration (AVI)

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

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

- Contraction property of $\mathcal{T} \Longrightarrow V_k \to V^*$.

- 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

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

- If $\mathcal{A}$ is a projection in L_∞-norm, then $\mathcal{A}$ is a non-expansion and the joint operator $\mathcal{A}\mathcal{T}$ is a contraction, which guarantees the existence of a unique fixed point $\tilde{V} = \mathcal{A}\mathcal{T}\tilde{V}$ 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:

$$Q_{k+1} = \mathcal{A}\mathcal{T}Q_k$$

  with $\mathcal{A}$ defined over S x 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

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

With features

$$\varphi_j : S \times A \to [0, L] \qquad \phi(s,a) = [\varphi_1(s,a) \ldots \varphi_d(s,a)]^T$$

- Each iteration, solve:

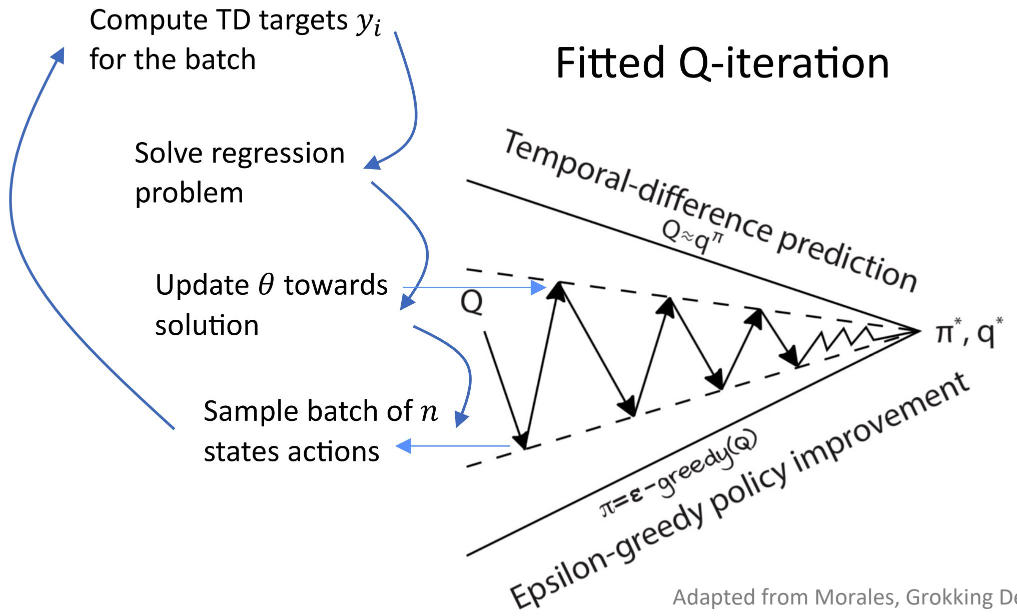
$$Q_{k+1} = \arg\min_{Q \in \mathcal{F}} ||Q - \mathcal{T} Q_k||_\mu^2$$

With μ a distribution over S.

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

Outline

Fitted Q-iteration (approximate Q-iteration)


Compute TD targets y_i for the batch
Solve regression problem
Update θ towards solution
Sample batch of n states actions
Fitted Q-iteration
Temporal-difference prediction
$Q \approx q^\pi$
Q
π^*, q^*
$\pi = \varepsilon\text{-greedy}(Q)$
Epsilon-greedy policy improvement

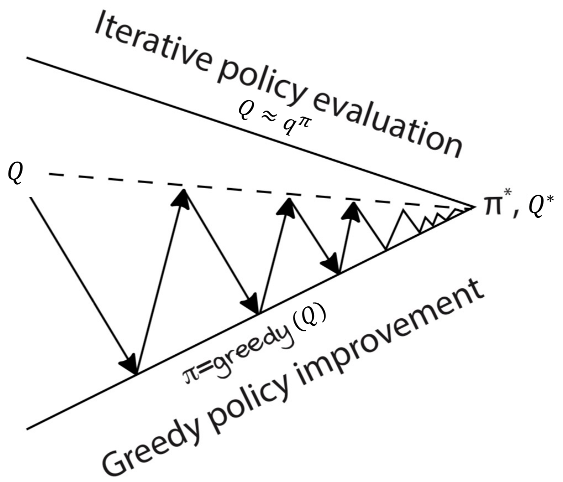
Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

Batch + approximate extension of Q-learning

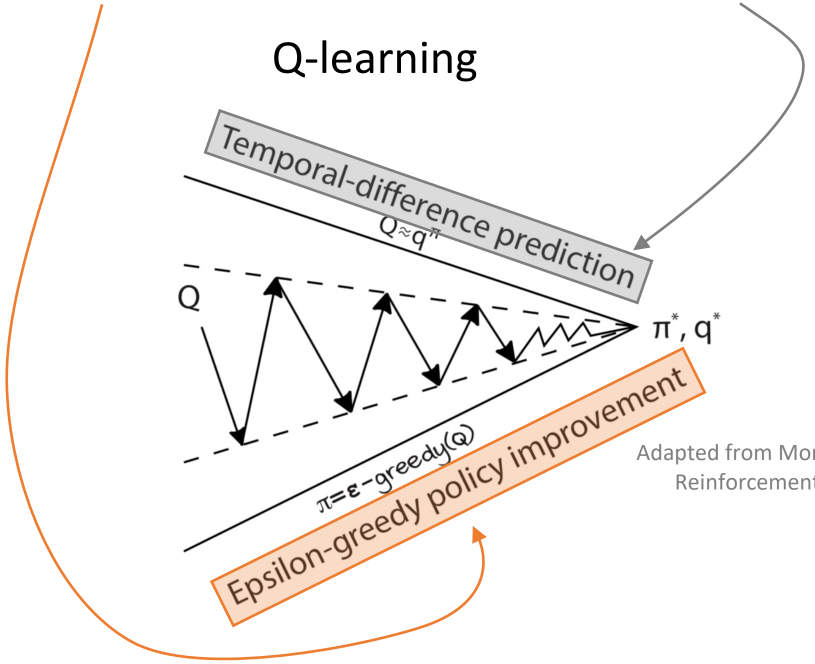
Recall: Q-learning

- Key idea: incrementally obtain new data and update Q function using the optimal Bellman equation (greedy)

Q-value iteration

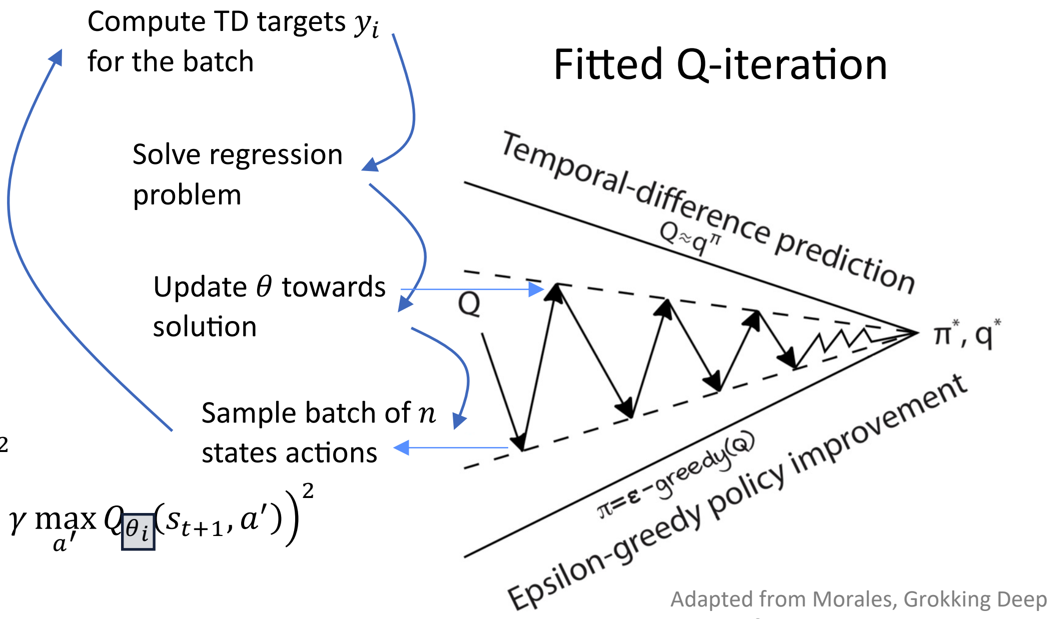


Q-learning


Temporal-difference prediction
Q≈q$^{\pi}$
Q
π^*, q*
π=ε-greedy(Q)
Epsilon-greedy policy improvement

Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

Fitted Q-iteration (approximate Q-iteration)

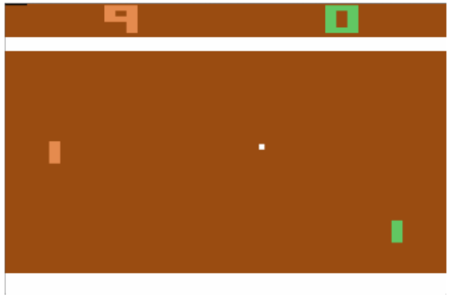

Compute TD targets y_i for the batch
Solve regression problem
Update θ towards solution
Sample batch of n states actions
Fitted Q-iteration
Temporal-difference prediction
Q≈q^π
Q
π*, q*
π=ε-greedy(Q)
Epsilon-greedy policy improvement

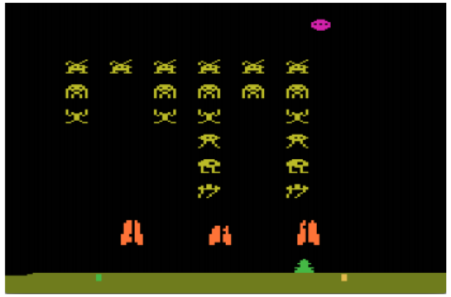
Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

- Run π, e.g., over a single trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n)$
- Q-iteration loss using bootstrapped target $\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta) = \sum_t (Q_\theta(s_t, a_t) - y_t)^2$
$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\boxed{\theta_i}}(s_{t+1}, a')\right)^2$$
- QL online update with learning rate α_i
$$\hat{\theta}_{i+1} = \hat{\theta}_i - \alpha_i \nabla_\theta \tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$ Not passing the gradient through this θ!
$$= \hat{\theta}_i - \alpha_i \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\boxed{\theta_i}}(s_{t+1}, a')\right) \nabla_\theta Q_\theta(s_t, a_t)$$

👎 Pseudo-gradient method. Here, it may diverge even with linear function approximation…

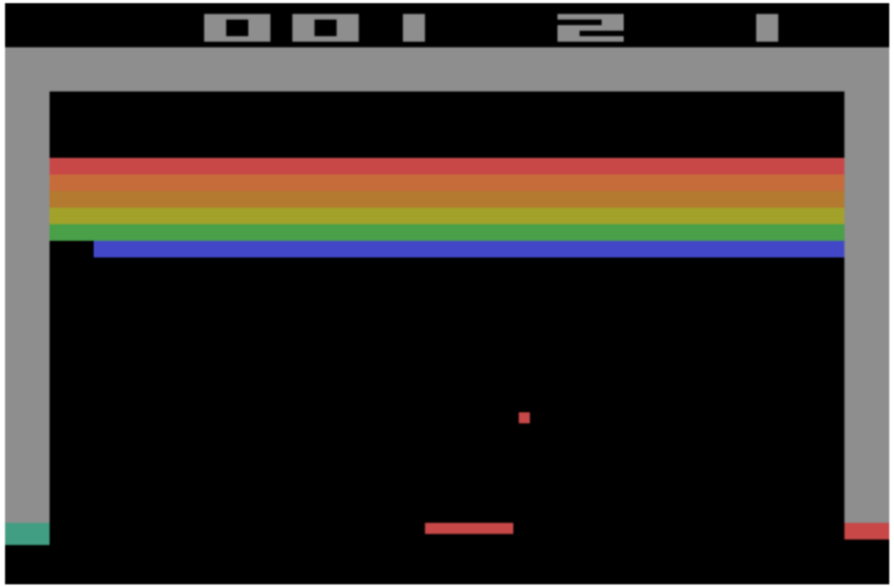
Fitted Q-iteration applied to ATARI Games





Atari

Image preprocessing: grey-scale, crop to 84x84

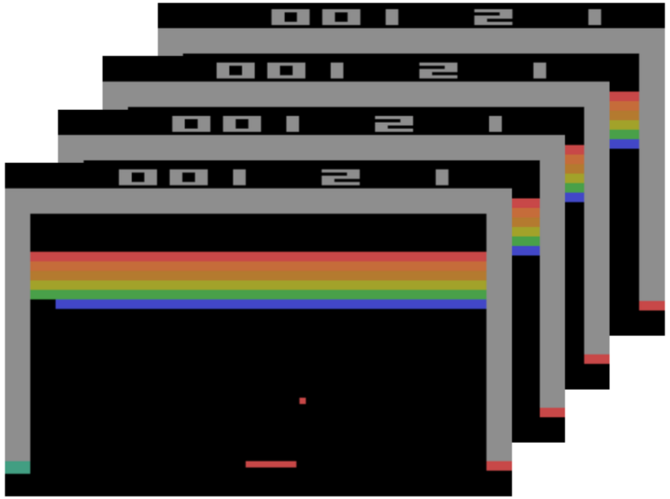
$\phi\Bigg($ 

$\Bigg)$

Atari

State definition: 4 last frames

$$s_t = \left\{ \right.$$

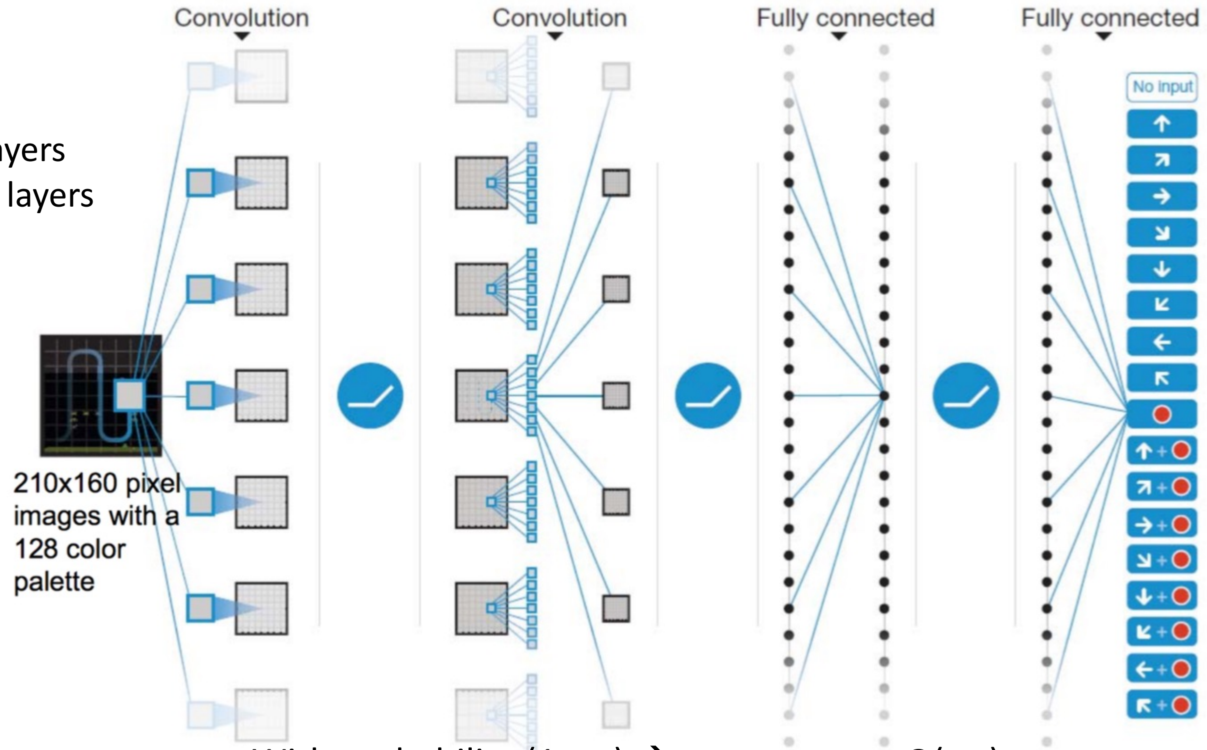


$$a_{t-1}, r_{t-1}, \\ \ldots \\ a_{t-4}, r_{t-4} \left. \right\}$$

Atari

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

Convolutional layers
Fully connected layers
…



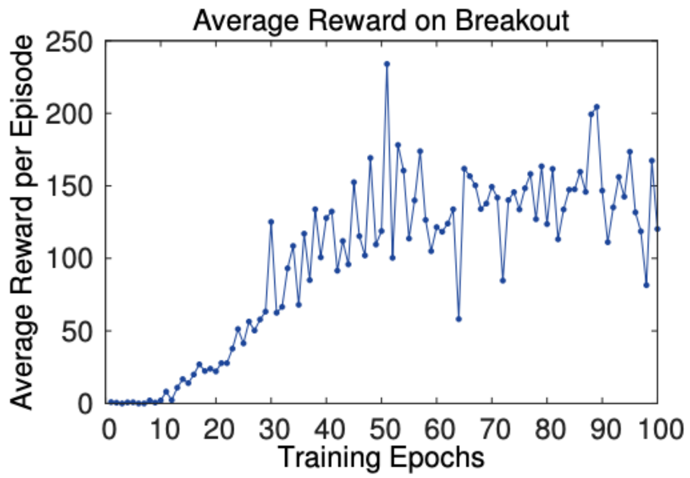
$Q(s, a_1)$

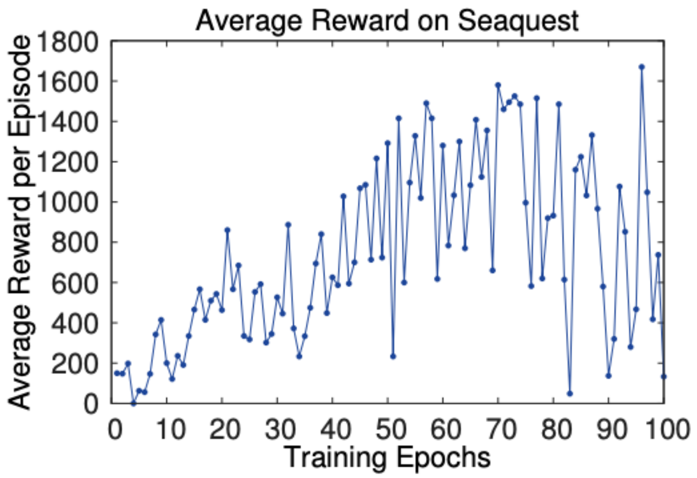
$Q(s, a_4)$

$Q(s, a_n)$

With probability $(1 - \varepsilon)$ → execute $\max_a Q(s,a)$
With probability ε → execute random action

Still doesn’t quite work. Why?



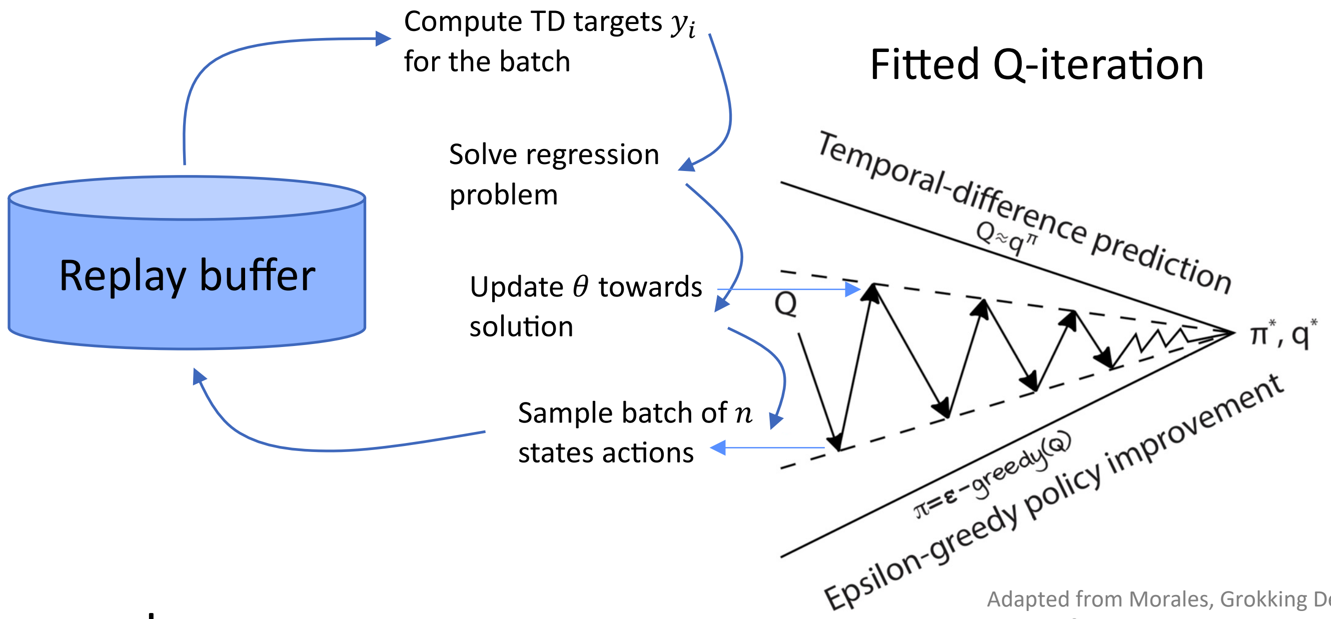


- Recall: Approximate QL as Pseudo-Gradient Descent
- Mathematically: Correlated samples. Violates i.i.d. assumption in supervised learning.
- Intuition: Unlike in supervised learning, in RL, the agent collects its own data. If that data is bad, then the result is bad too (and may make future collection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

Outline

1. Value function approximation
2. **Approximate value-based algorithms**
   a. Policy evaluation
   b. Policy learning: approximate value iteration
   c. Policy learning: fitted Q iteration
   d. **Policy learning: deep Q-network**

Solution: increase data diversity!


Compute TD targets y_i for the batch
Solve regression problem
Update θ towards solution
Sample batch of n states actions
Replay buffer
Fitted Q-iteration
Temporal-difference prediction
$Q \approx q^{\pi}$
Q
π^*, q^*
$\pi = \varepsilon\text{-greedy}(Q)$
Epsilon-greedy policy improvement

Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

- 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

$$\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$

$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a') \right)^2$$

- Iteration $i + 1$

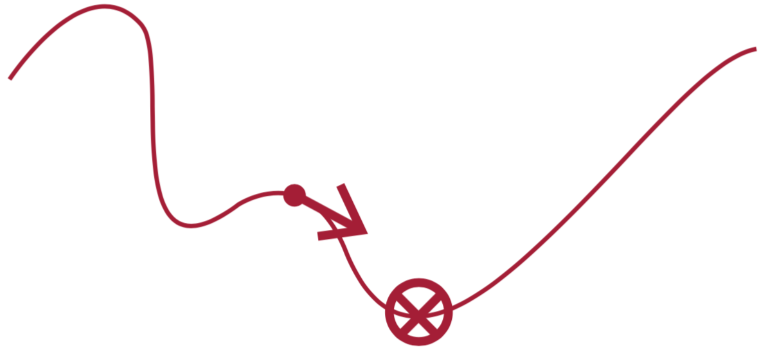
$$\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$

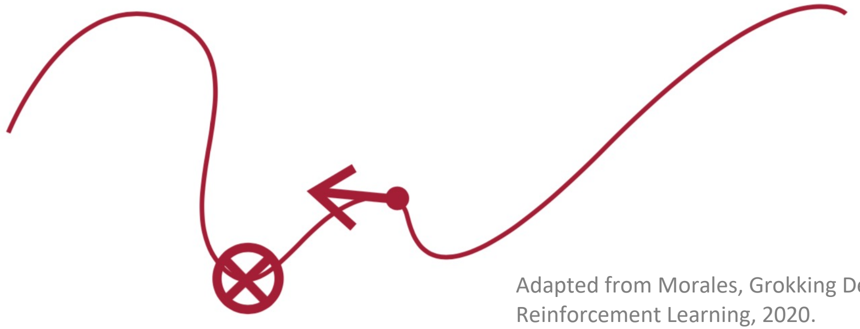
$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_{i+1}}(s_{t+1}, a') \right)^2$$

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

$$\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$

$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\bar{\theta}}(s_{t+1}, a') \right)^2$$

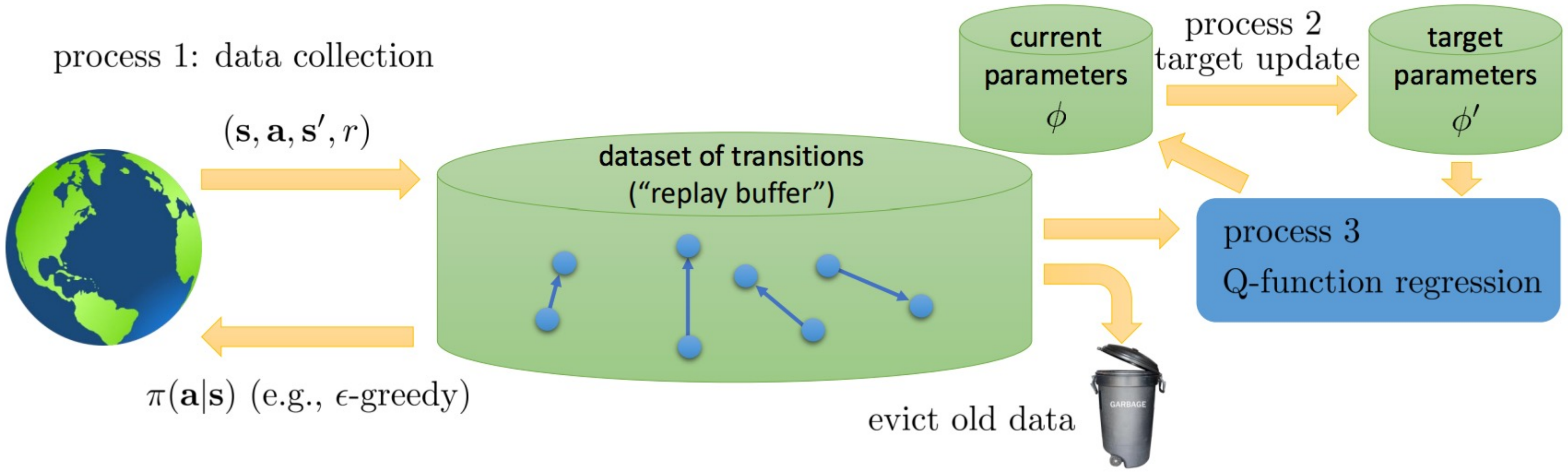




Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

“Target network”

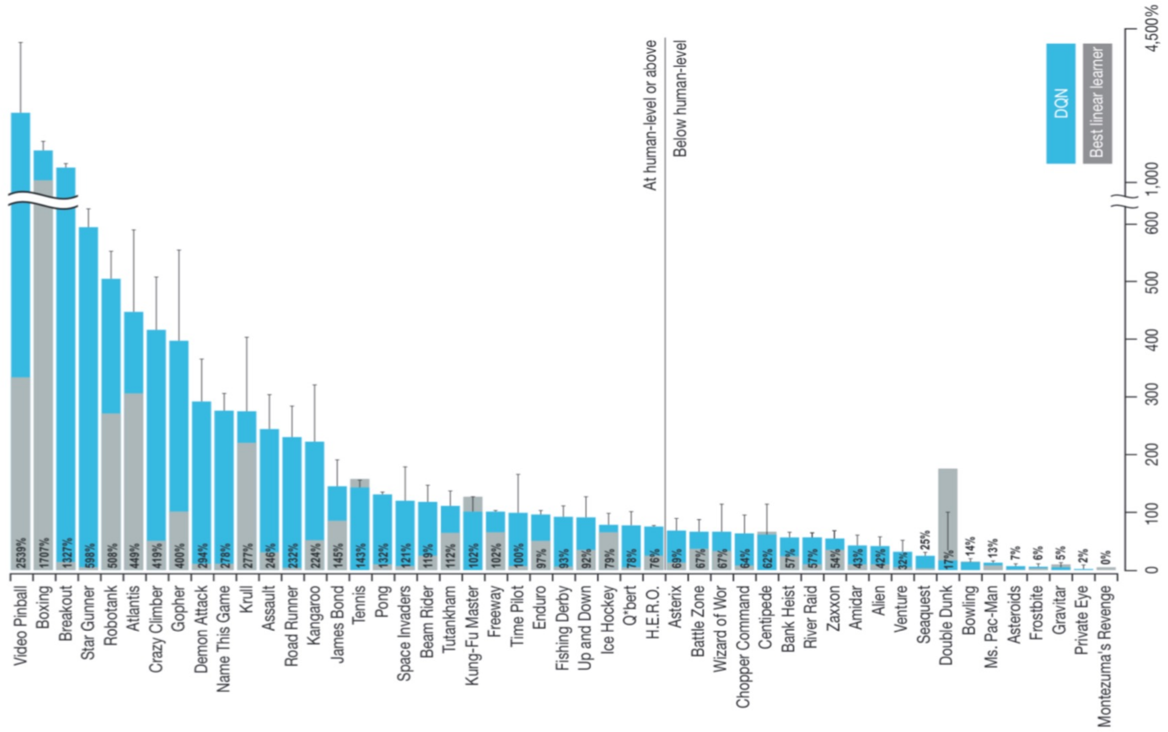
DQN algorithm



Adapted from Sergey Levine

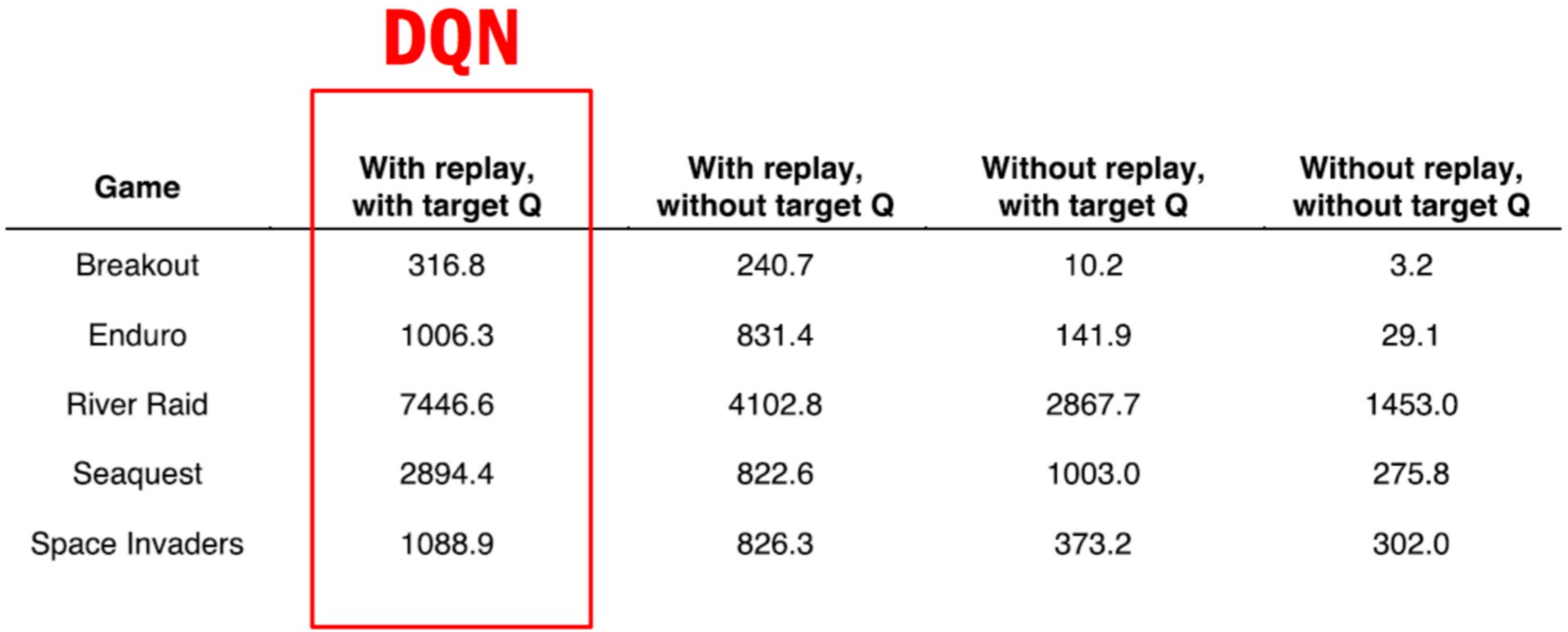
DQN – Atari

Performance



DQN – Atari

Ablation

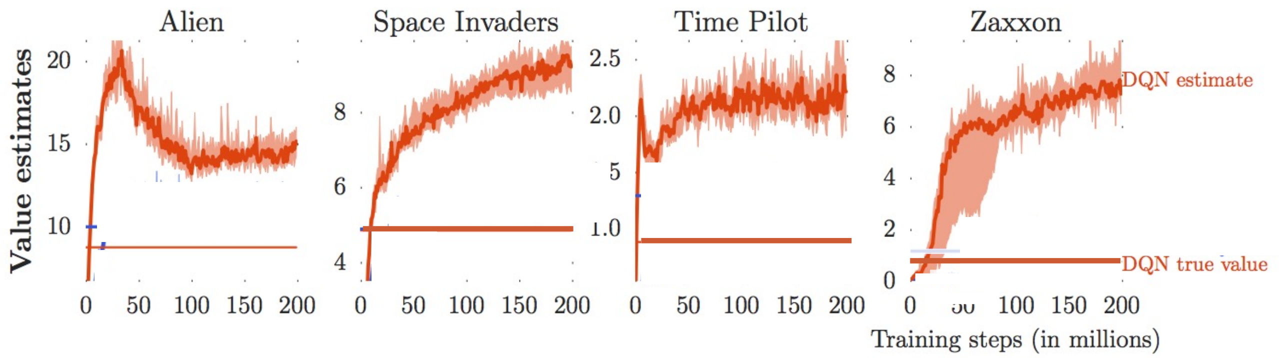


DQN

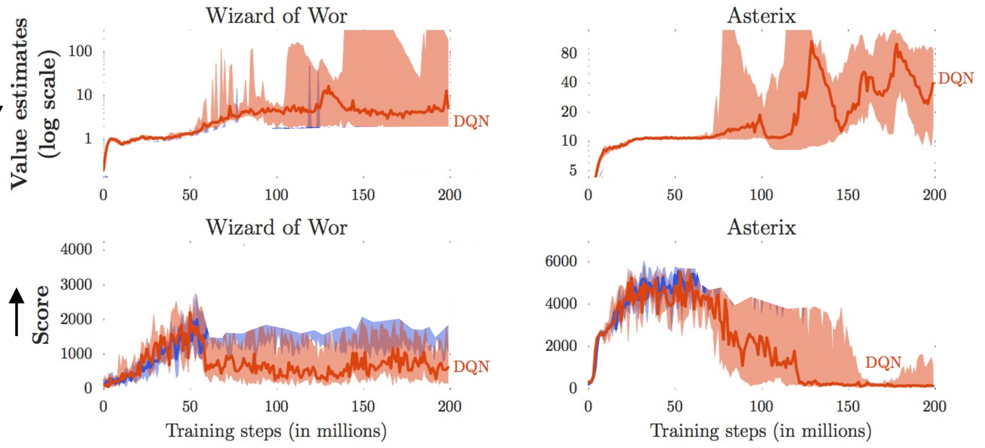
Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

Limitations of DQN

- DQN over-estimates Q values



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



Deep Reinforcement Learning with Double Q-Learning, Hasselt et al., 2015

Over-estimation in DQN

Recall: $\max_a Q(s,a)$ is part of computing the TD targets: $r_t + \gamma \max_a Q(s_{t+1}, a)$

- Consider

$$\max_a Q(s,a)$$

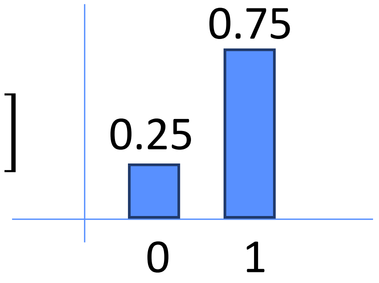
Note: In practice, $\hat{Q}$ is usually a deterministic function. Here we use a random variable to represent uncertainty in the actions & randomness in the trajectory / MDP.

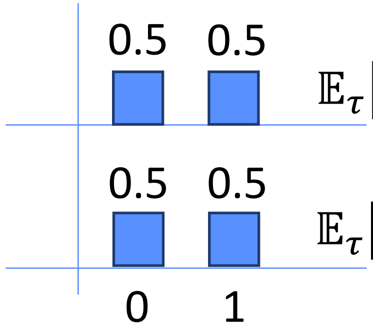
- Over-estimation issue:

$$\mathbb{E}_\tau\left[\max_a \hat{Q}(s,a)\right] \geq \max_a \mathbb{E}_\tau\left[\hat{Q}(s,a)\right]$$

- Example 1: let $\hat{Q}(s,a) \sim \mathrm{Ber}(0.5)$ for two actions, a_1, a_2

$$0.75 \leftarrow \mathbb{E}_\tau\left[\max_a \hat{Q}(s,a)\right]$$





$$\mathbb{E}_\tau\left[\hat{Q}(s,a_1)\right] \to 0.5$$

$$\mathbb{E}_\tau\left[\hat{Q}(s,a_2)\right] \to 0.5$$

- Issue 1: Sampling $\max_a \hat{Q}(s,a)$ will over-estimate Q-values

Over-estimation in DQN

- Consider

$$\max_a Q(s,a)$$

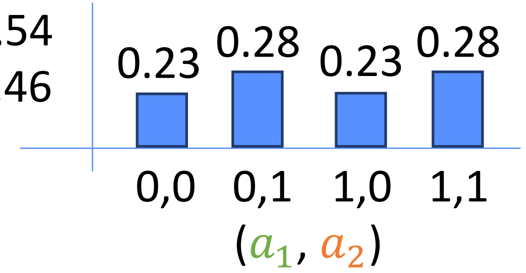
- Over-estimation issue:

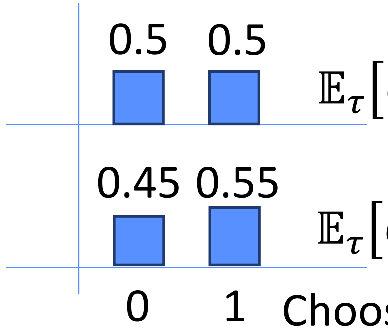
$$\mathbb{E}_\tau\left[\max_a \hat{Q}(s,a)\right] \geq \max_a \mathbb{E}_\tau\left[\hat{Q}(s,a)\right]$$

- Example 2: let $\hat{Q}(s,a) \sim \mathrm{Ber}(0.5)$ for a_1, $\mathrm{Ber}(0.55)$ for a_2

Note: In practice, $\hat{Q}$ is usually a deterministic function. Here we use a random variable to represent uncertainty in the actions & randomness in the trajectory / MDP.

Choose:
- a_2 with probability 0.54
- a_1 with probability 0.46





$\mathbb{E}_\tau[\hat{Q}(s,a_1)] \to 0.5$

$\mathbb{E}_\tau[\hat{Q}(s,a_2)] \to 0.55$

Choose: a_2 (with probability 1)

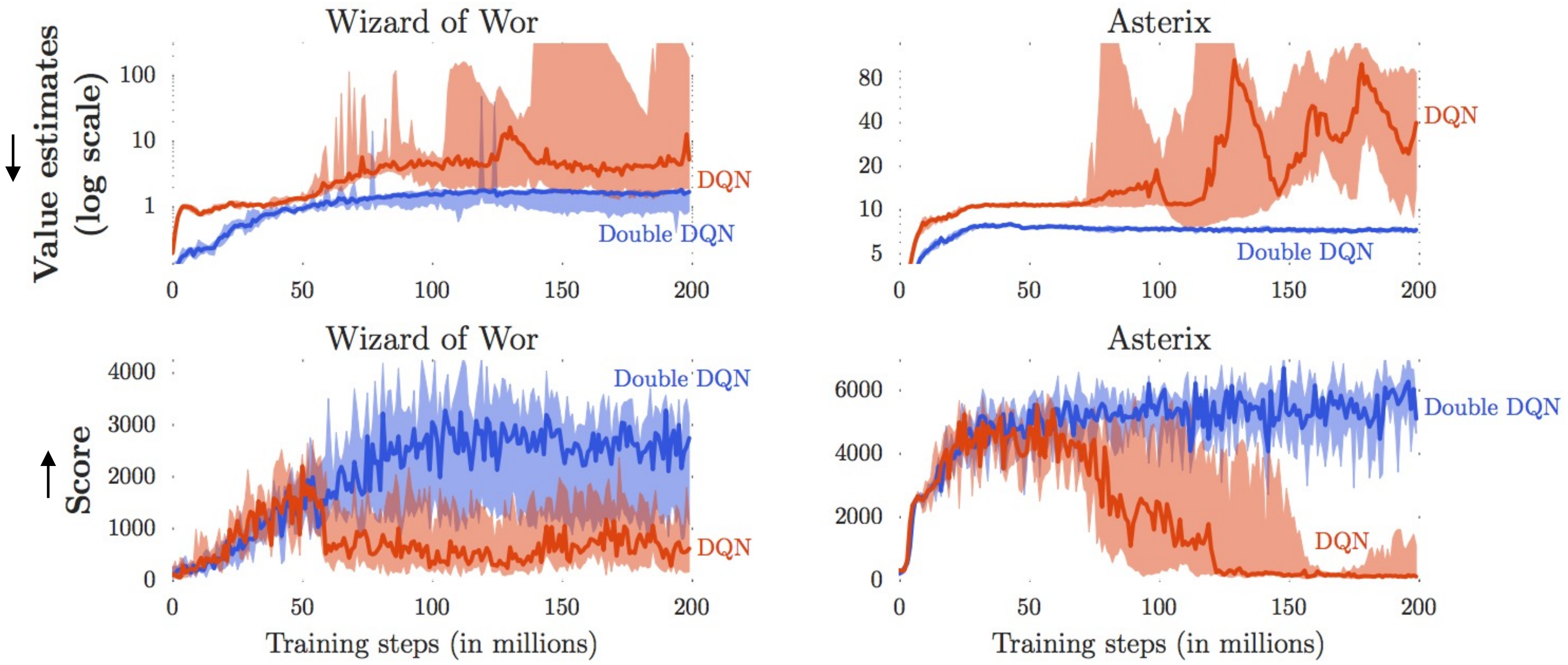
- Issue 2: Actions with lower expected Q-values will often be selected

Double DQN

- Solution: "re-sample" $\hat{Q}$ of the action you think is best. In expectation, the Q-value will be correct (not overestimated).
- But how? In practice, $\hat{Q}$ is deterministic. So use another Q-function.
- Double DQN
  - Implementation: Use two networks θ_1 and θ_2 (hence "double")
    - In practice: Can combine re-use the Target network as the 2nd Q-network.
  - Compute $a_{\max,t} = \arg\max_{a'} Q_{\theta_2}(s_{t+1}, a')$
  - Update
    $$\theta_1 \leftarrow \theta_1 - \alpha\left(Q_{\theta_1}(s_t, a_t) - r_t - \gamma Q_{\theta_1}\left(s_{t+1}, a_{\max,t}\right)\right)\nabla_\theta Q_{\theta_1}(s_t, a_t)$$
  - Alternate between θ_1 and θ_2
- Remark: Double Q-learning is the tabular version of double DQN. Under the same conditions to Q-learning, double Q-learning converges a.s. to Q^*.

Hasselt. Double Q-learning, NIPS 2010.

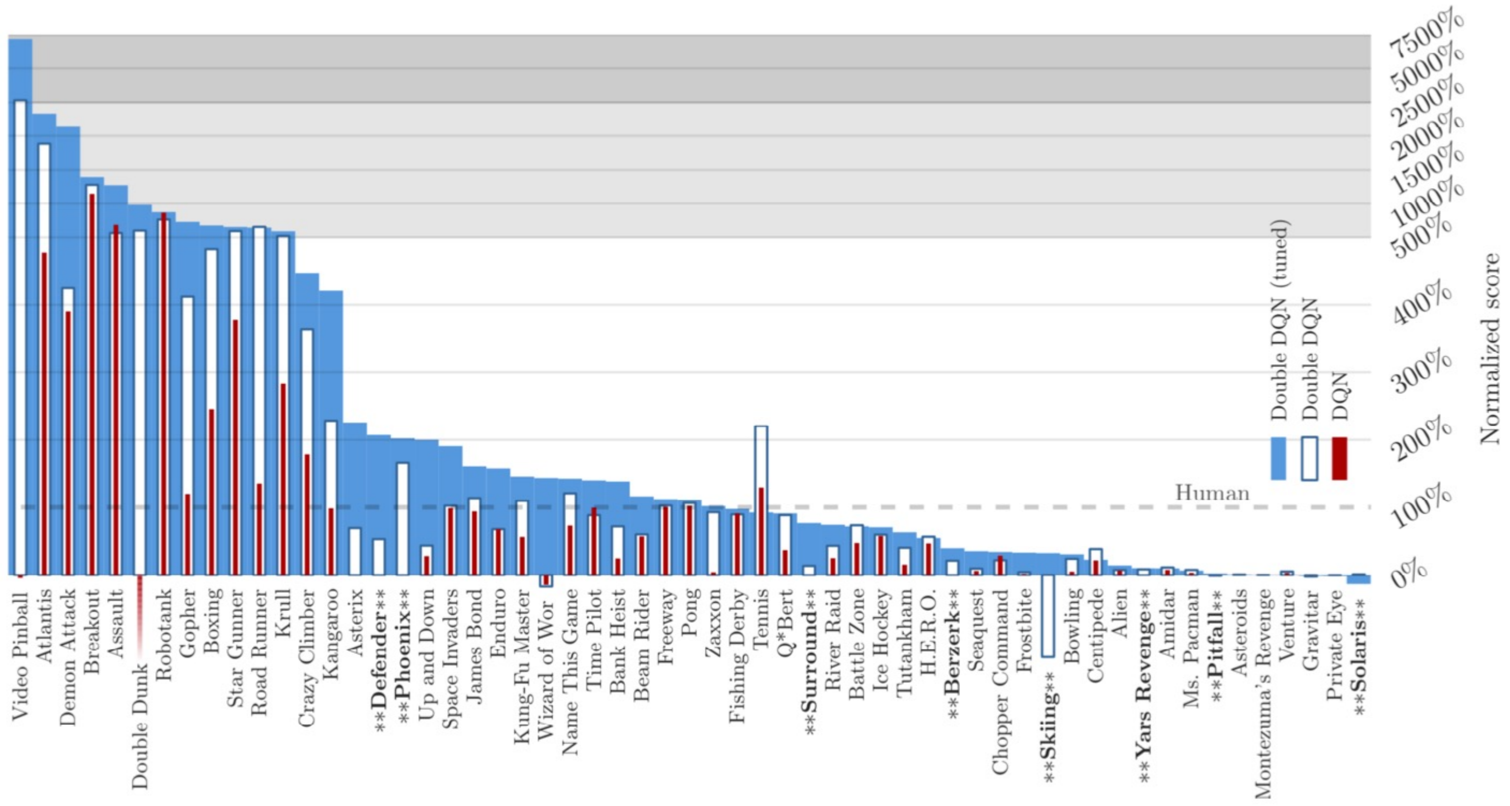
Double DQN - Atari



Deep Reinforcement Learning with Double Q-Learning, Hasselt et al., 2015

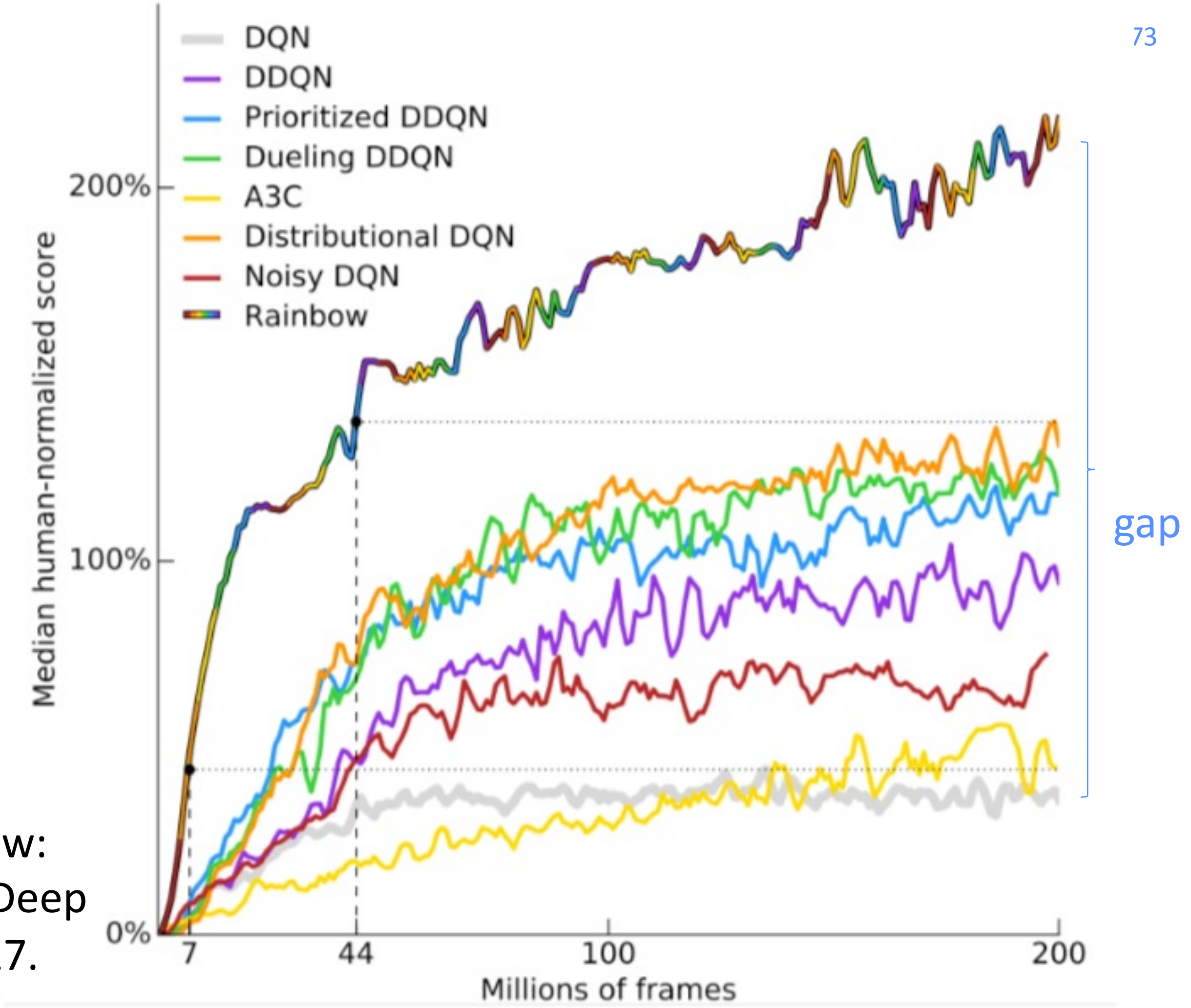
Double DQN - Atari

~3 lines of code
Always worth trying



van Hasselt, Guez, Silver, 2015

Rainbow DQN



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

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

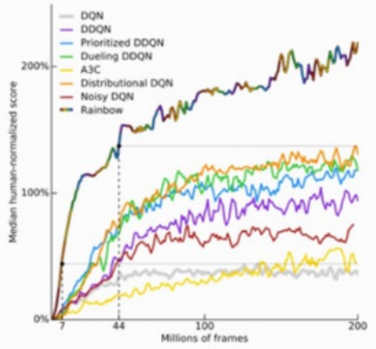


Figure 1: **Median human-normalized performance** across 57 Atari games. We compare our integrated agent (rainbow-colored) to DQN (grey) and six published baselines. Note that we match DQN's best performance after 7M frames, surpass any baseline within 44M frames, and reach substantially improved final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

The cost of Rainbow:

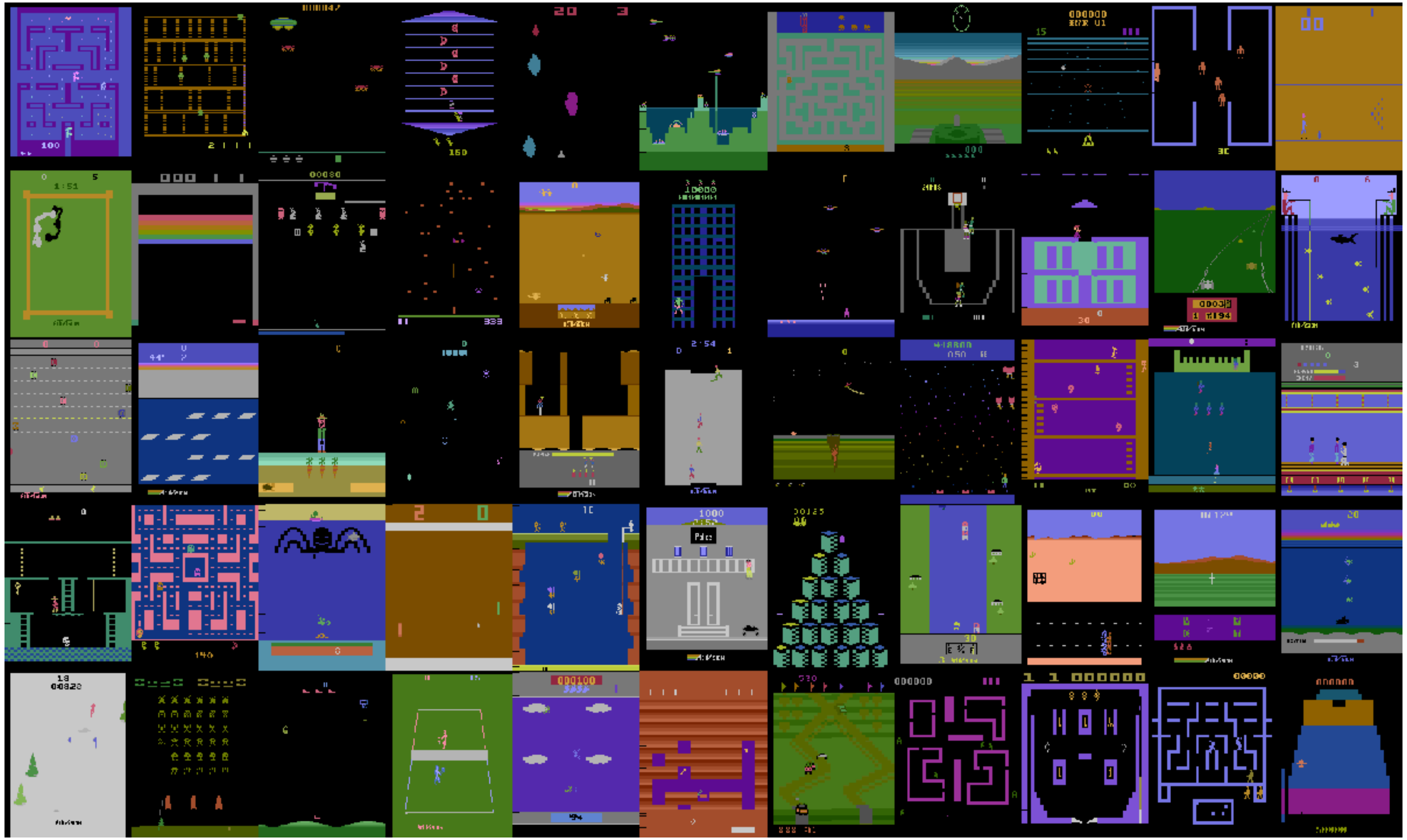
- ~5 days to train a game on a P100
- 57 games
- 5 independent seeds
- P100 costs about US$6000

Lower-bound of cost:

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

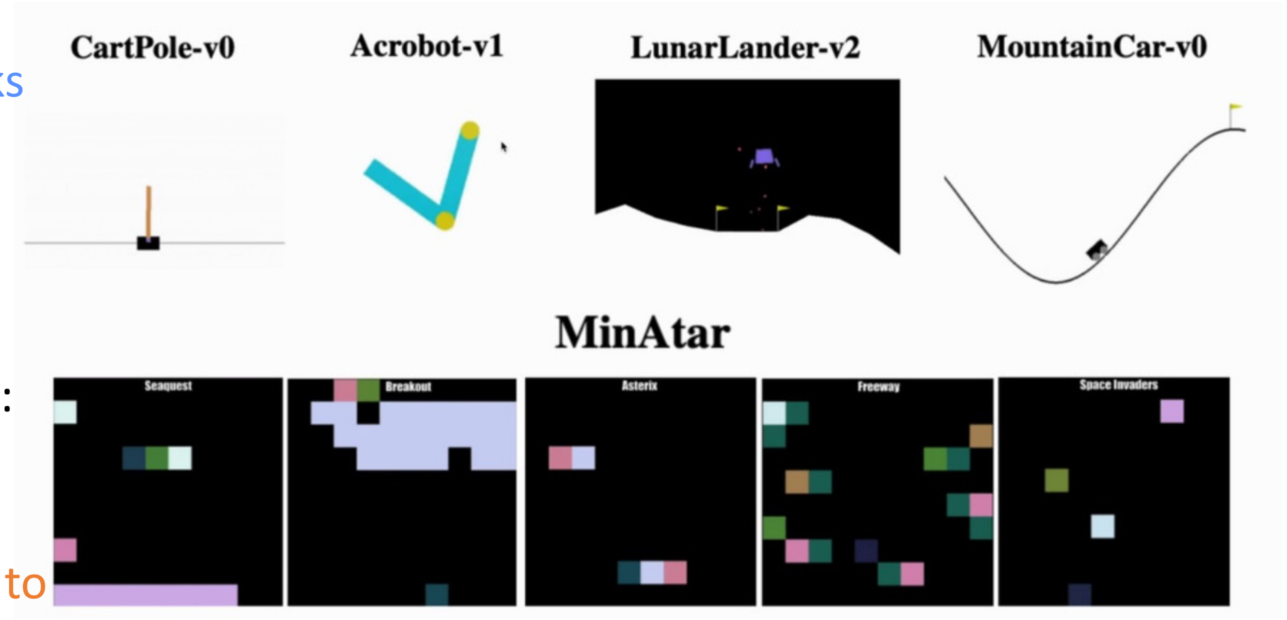
Arcade Learning Environment (ALE)

50+ Atari 2600 games



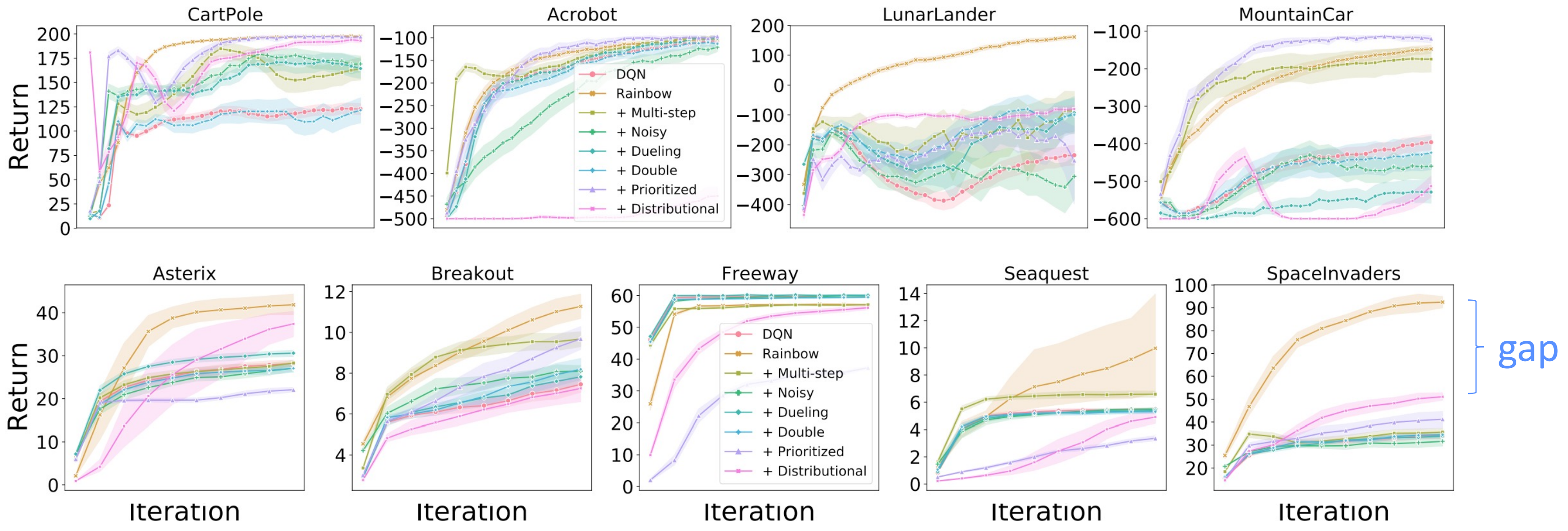
Revisiting Rainbow: more is not always better

- 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



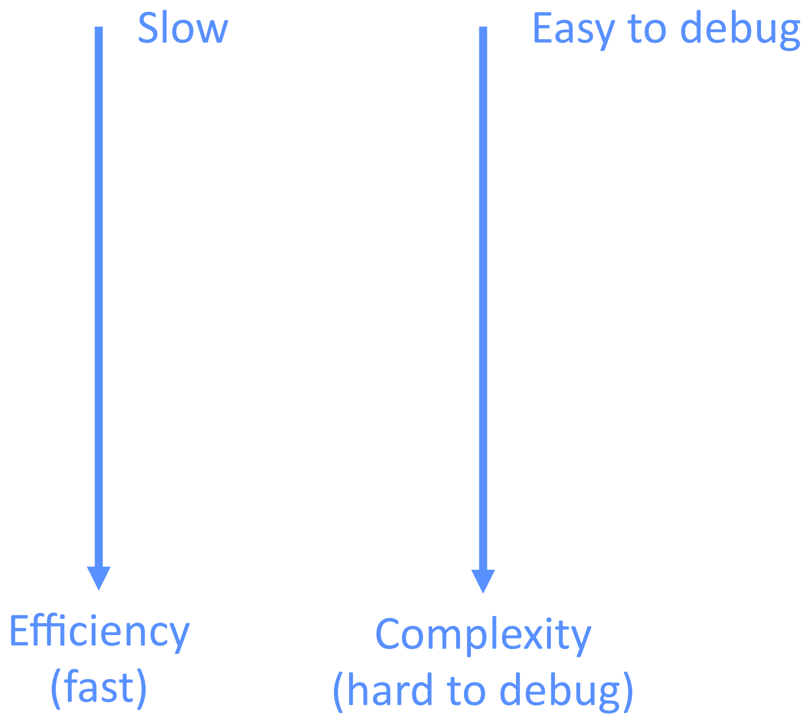
Revisiting Rainbow: more is not always better

- Rainbow still >> DQN



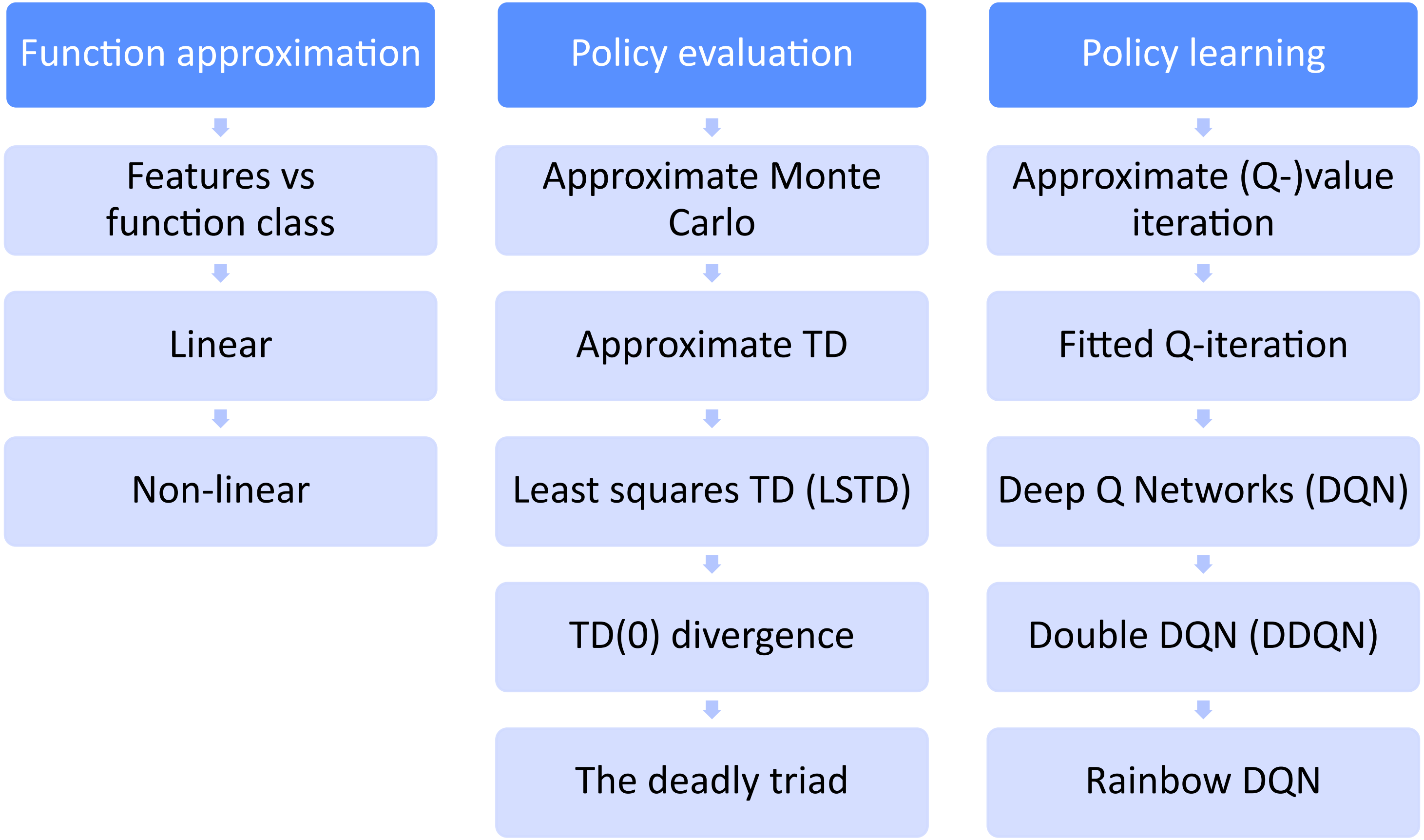
A menu of value-based (RL) algorithms

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



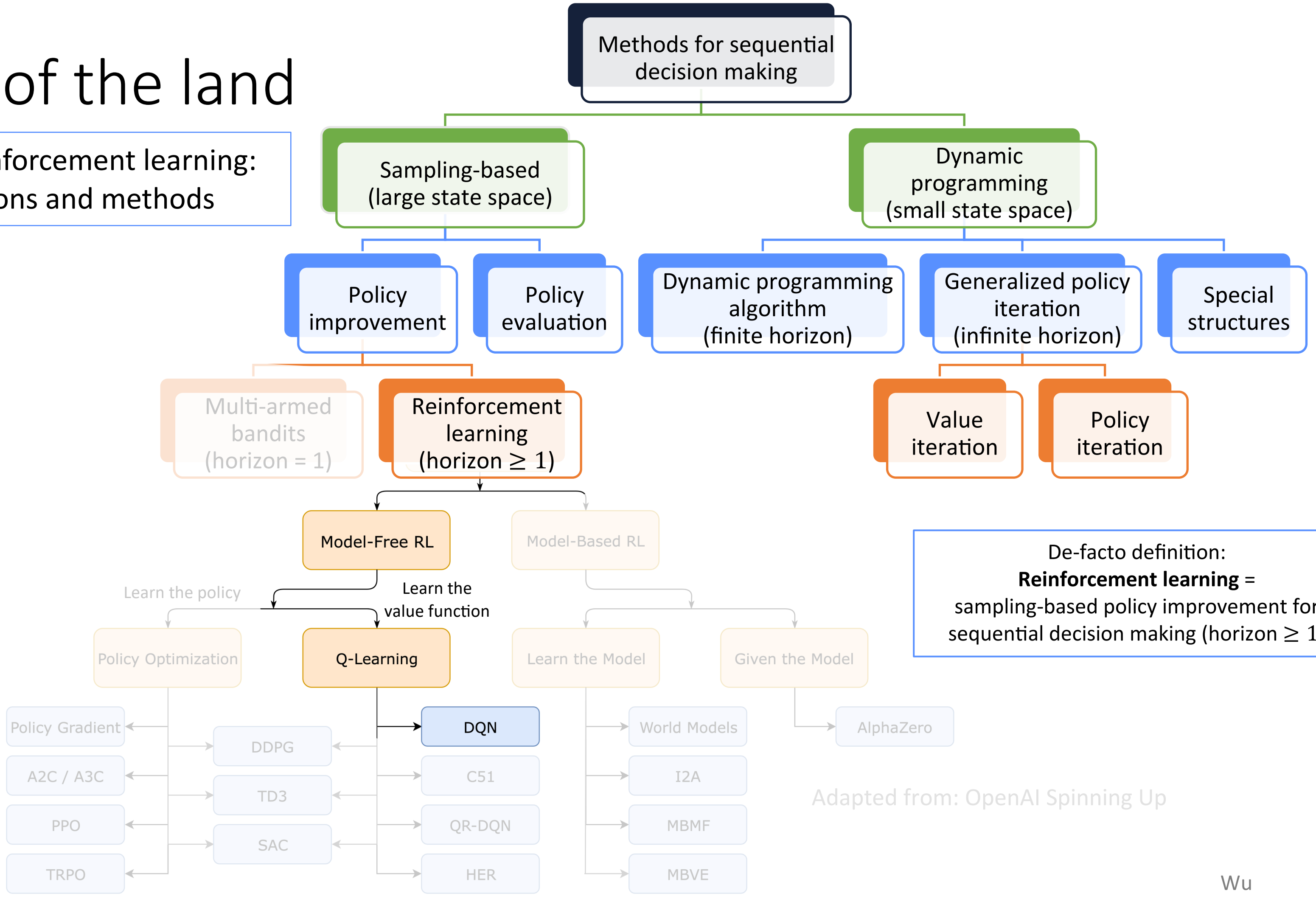
Tip: Use the simplest algorithm the solves your problem

From Q-learning to DQN



Lay of the land

6.7920: Reinforcement learning: foundations and methods



De-facto definition:
Reinforcement learning =
sampling-based policy improvement for
sequential decision making (horizon ≥ 1)

Adapted from: OpenAI Spinning Up

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 Linear TD (Prop 6.5)

And theorem for the incremental version

Proof: Linear TD ($\lambda = 0 \to \alpha_0 = \gamma$)

First, want to show that $\Pi^{\rho^\pi}\mathcal{T}^\pi$ is a contraction in L_{2,ρ_π} with a unique fixed point.

- The transition matrix induced by π does not increase the (weighted) norm:
  - $\|P^\pi V\|_{\rho^\pi}^2 = \sum_s \rho^\pi(s)\left(\sum_{s'} p(s'|s,\pi(s))V(s')\right)^2$
    $$\le \sum_s \rho^\pi(s) \sum_{s'} p(s'|s,\pi(s))V(s')^2$$
    $$= \sum_{s'} \rho^\pi(s')V(s')^2 = \|V\|_{\rho^\pi}^2$$
  - The second inequality follows the Jensen's inequality.
  - The third equality holds because ρ^π is a stationary distribution.
- Then it immediately follows that $\mathcal{T}^\pi$ is a contraction in L_{2,ρ_π}, i.e.,
  $$\|\mathcal{T}^\pi V_1 - \mathcal{T}^\pi V_2\|_{\rho_\pi} = \gamma\|P^\pi(V_1 - V_2)\|_{\rho_\pi} \le \gamma\|V_1 - V_2\|_{\rho_\pi}$$
- It can be shown that Π^{ρ^π} is a non-expansion.
- $\rightarrow$ unique fixed point $V_{TD} = \Pi^{\rho^\pi}\mathcal{T}^\pi V_{TD}$.

Proof: Linear TD ($\lambda = 0 \to \alpha_0 = \gamma$)

Now, let's consider the error:

- By Pythagorean theorem, we have:

$$\|V^\pi - V_{TD}\|_{\rho^\pi}^2$$
$$= \left\|V^\pi - \Pi^{\rho^\pi} V^\pi\right\|_{\rho^\pi}^2 + \left\|\Pi^{\rho^\pi} V^\pi - V_{TD}\right\|_{\rho^\pi}^2$$

- But:

$$\left\|\Pi^{\rho^\pi} V^\pi - V_{TD}\right\|_{\rho^\pi}^2 = \left\|\Pi^{\rho^\pi} V^\pi - \Pi^{\rho^\pi} \mathcal{T}^\pi V_{TD}\right\|_{\rho^\pi}^2$$
$$\leq \|\mathcal{T}^\pi V^\pi - \mathcal{T}^\pi V_{TD}\|_{\rho^\pi}^2 \leq \gamma^2 \|V^\pi - V_{TD}\|_{\rho^\pi}^2$$

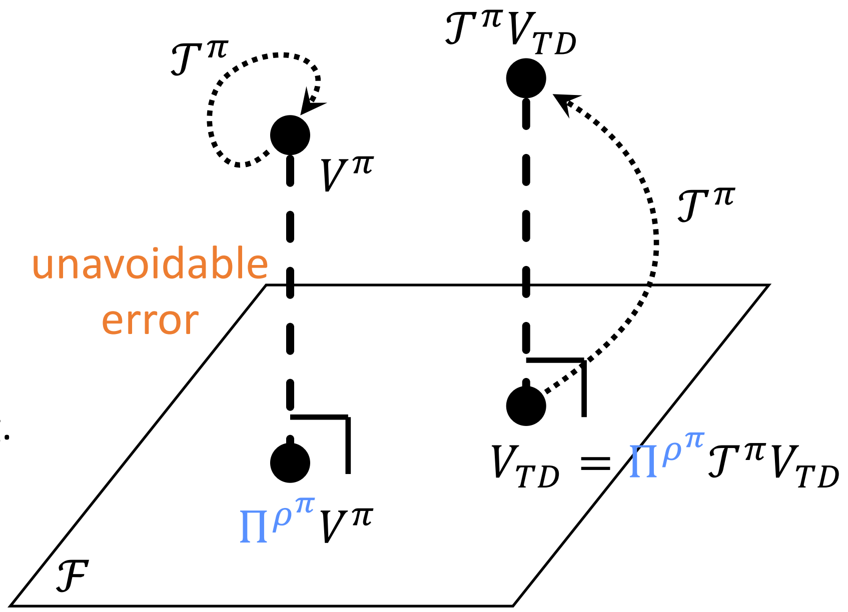
- Thus

$$\|V^\pi - V_{TD}\|_{\rho^\pi}^2$$
$$\leq \left\|V^\pi - \Pi^{\rho^\pi} V^\pi\right\|_{\rho^\pi}^2 + \gamma^2 \|V^\pi - V_{TD}\|_{\rho^\pi}^2$$

- Which corresponds to the result after reordering.

■

$$V_{TD} = \Phi\theta^* = \Pi^{\rho^\pi} \mathcal{T}^\pi \Phi\theta^*$$
$$= \Pi^{\rho^\pi} \mathcal{T}^\pi V_{TD}$$



Incremental Linear TD

Theorem (Tsitsiklis and Van Roy, 1996; c.f. NDP, Prop 6.5)

Let the learning rate η_t satisfy $\sum_{t\geq 0} \eta_t = \infty$, and $\sum_{t\geq 0} \eta_t^2 < \infty$.

We assume that there exists a distribution ρ over $\mathcal{S}$ such that $\forall s, s' \in \mathcal{S}$, $\lim_{t\to\infty} P(s_t = s' | s_0 = s) = \rho(s') > 0$ and that the features $(\phi_i)_{1\leq k\leq K\leq |\mathcal{S}|}$ are linearly independent. Let the updates be given by:

$$\theta_{t+1} = \theta_t + \eta_t \delta_t \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \phi(s_k)$$

Then there exists a fixed θ^* such that $\lim_{t\to\infty} \theta_t = \theta^*$. Furthermore, we obtain:

$$\|V_{\theta^*} - V^\pi\|_{2,\rho} \leq \frac{1-\lambda\gamma}{1-\gamma} \inf_\theta \|V_\theta - V^\pi\|_{2,\rho}$$

☞ See NDP §6.3.3 for detailed analysis. Need to cope with updates that worsen value function. Noise is NOT conditionally mean zero. Leverage stochastic approximation results for Markovian noise (NDP §4.4).

Reference: Detailed proof for Approximate Value Iteration

And performance loss lemma

Proof: Approximate Value Iteration

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

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

then

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

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

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

□

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

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

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

Proof: Approximation Error to Performance Loss

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

■