Planning and Optimal Control

7. Action Value Learning (Q Learning)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

		A. Models for Sequential Data
Tue. 22.10.	(1)	1. Markov Models
Tue. 29.10.	(2)	2. Hidden Markov Models
Tue. 5.11.	(3)	3. State Space Models
Tue. 12.11.	(4)	3b. (ctd.)
		B. Models for Sequential Decisions
Tue. 19.11.	(5)	1. Markov Decision Processes
Tue. 26.11.	(6)	1b. (ctd.)
Tue. 3.12.	(7)	1c. (ctd.)
Tue. 10.12.	(8)	2. Monte Carlo and Temporal Difference Methods
Tue. 17.12.	(9)	3. Q Learning
Tue. 24.12.	—	— *Christmas Break* —
Tue. 7.1.	(10)	4. Policy Gradient Methods
Tue. 14.1.	(11)	tba
Tue. 21.1.	(12)	tba
Tue. 28.1.	(13)	8. Reinforcement Learning for Games
Tue. 4.2.	(14)	Q&A

Outline

1. Basic Q Learning
2. Action Value Models $\hat{Q}$
3. Experience Replay / Relabeling
4. Deep Q Learning

Outline

Brief Review

what to learn / **target**		
value function $V^\pi : S \to \mathbb{R}$		
action value function $Q^\pi : S \times A \to \mathbb{R}$		

Brief Review

what to learn / **target**	what to learn from / **data**: episodes (Monte Carlo)	what to learn from / **data**: transitions (temporal differences)
value function $V^\pi : S \to \mathbb{R}$		
action value function $Q^\pi : S \times A \to \mathbb{R}$		

Brief Review

what to learn / **target**	what to learn from / **data**: episodes (Monte Carlo)	what to learn from / **data**: transitions (temporal differences)
value function $V^\pi : S \to \mathbb{R}$	Monte Carlo	Temporal Differences TD(0)
action value function $Q^\pi : S \times A \to \mathbb{R}$	Monte Carlo	SARSA

Brief Review

what to learn / **target**	what to learn from / **data**: episodes (Monte Carlo)	what to learn from / **data**: transitions (temporal differences)
value function $V^\pi : S \to \mathbb{R}$	Monte Carlo	Temporal Differences TD(0)
action value function $Q^\pi : S \times A \to \mathbb{R}$	Monte Carlo	SARSA
optimal action value function $Q^* : S \times A \to \mathbb{R}$		Q-Learning

Brief Review / SARSA

- learn the action value function Q^π of the generating policy ("**on policy**"):

$$\hat{Q}_{s_t,a_t} = \hat{Q}_{s_t,a_t} + \alpha(r_t + \gamma \hat{Q}_{s_{t+1},a_{t+1}} - \hat{Q}_{s_t,a_t})$$

- can be used with any policy π to learn its action value function Q^π.
- requires next action a_{t+1} to update $\hat{Q}_{s_t,a_t}$.
- can be used with non-stationary policies such as **ϵ-greedy policy**:

$$\begin{aligned}\pi(s,a;\hat{Q},\epsilon) &:= (1-\epsilon)\,\pi_{\text{greedy}}(s,a;\hat{Q}) + \epsilon\,\pi_{\text{uniform}}(s,a), \quad \epsilon \in [0,1] \\ &= (1-\epsilon)\,\mathbb{I}(a = \arg\max_{a \in A} \hat{Q}(s,a)) + \epsilon\,\frac{1}{|A|}\end{aligned}$$

with $\pi \to \pi^*$ for $\epsilon \to 0$,
to learn the optimal action value function Q^* and optimal policy π^*.

Update Rule

SARSA update rule:

$$\hat{Q}_{s_t,a_t} = \hat{Q}_{s_t,a_t} + \alpha(r_t + \gamma \hat{Q}_{s_{t+1},a_{t+1}} - \hat{Q}_{s_t,a_t})$$

Q-Learning update rule:

$$\hat{Q}_{s_t,a_t} = \hat{Q}_{s_t,a_t} + \alpha(r_t + \gamma \max_a \hat{Q}_{s_{t+1},a} - \hat{Q}_{s_t,a_t})$$

- does not require next action a_{t+1}
- does not learn the action value function Q^π of the generating policy ! ("**off policy**")
- learns the action value function of the policy that
  - takes optimal action at next time
    — based on current estimates of the action value function
- used with non-stationary policies such as ϵ-**greedy policy**: learns optimal action value function Q^*.

Convergence

If

- there are finite many states and actions,
- the generating policy visits each state/action pair an infinite number of times,
- the learning rates are slowly diminishing ($\sum_k \alpha_k(s,a) = \infty, \quad \sum_k \alpha_k(s,a)^2 < \infty$) and
- $\gamma < 1$ or (if $\gamma = 1$) there exists an absorbing state with zero reward for any policy.

then the Q-learning estimates converge almost surely to the optimal action value function Q^*.

Note: E.g., learning rates $\alpha_k(s,a) = 1/k$.

Q-Learning

learn-opt-policy-discounted-q-learning$(S, A, \gamma, s_{\text{term}}, N, \pi, q_0, \alpha)$:

$\hat{Q} := (q_0)_{s \in S, a \in A}$
for $n := 1, \ldots, N$:
  $s :=$ new_process()
  while $s \neq s_{\text{term}}$:
    $a := \pi(s)$
    $(r, s') :=$ execute_action(s, a)
    $\hat{Q}_{s,a} := \hat{Q}_{s,a} + \alpha_n(r + \gamma \max_{a' \in A} \hat{Q}_{s',a'} - \hat{Q}_{s,a})$
    $s := s'$
for $s \in S$:
  $\hat{\pi}^*_s := \arg\max_{a \in A} \hat{Q}(s, a)$
return $\hat{Q}, \hat{\pi}^*$

where

- s_{term} **terminal state** with zero reward.
- π **generating policy**
- $q_0 \in \mathbb{R}$ initial value of all state/action pairs.
- α_k **learning rate** for update step k, e.g., $\alpha_k := 1/k$.
- new_process() sets up a new process.
- execute_action(s, a) executes action a in process in state s.

Outline

Action Value Models Q

- from an ML perspective, the action value function Q can be understood as a **regression model**:

$$Q : S \times A \to \mathbb{R}$$

  - with predictors s and a
  - for nominal states and actions:
    - constant model: just a number for every pair (s, a).
    - a factorization model likely would make better use of the data?
  - for **structured states and actions**:
    - a proper regression model for values regressed on state and action properties.
  - as for using Q to choose actions, $Q(s, a)$ for all competing actions a have to be computed, the model could be represented as:

$$Q : S \to \mathbb{R}^A$$

    - with structured input and
    - with structured output

Action Value Models Q / Learning

- Updating the action value model Q:
  - constant model:

8
$$\begin{aligned}\hat{Q}_{s,a} &:= \hat{Q}_{s,a} + \alpha_n(r + \gamma \max_{a' \in A} \hat{Q}_{s',a'} - \hat{Q}_{s,a}) \\ &= (1 - \alpha_n)\hat{Q}_{s,a} + \alpha_n(r + \gamma \max_{a' \in A} \hat{Q}_{s',a'})\end{aligned}$$

  - general model:

8
$$\mathcal{D} := \mathcal{D} \cup \{(s, a, r + \gamma \max_{a' \in A} \hat{Q}_{s',a'})\}$$
9
$$\hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D})$$

    - called **experience replay** in the RL literature

- if the generating policy π depends on $\hat{Q}$,
  learning can be accelerated by discounting older data.
  - e.g., with a discounting case weight.
  - or simply by forgetting / removing.

Outline

Action Value Models Q / Relabeling (1/2)

1. label with data generating $\hat{Q}$:

8 $\mathcal{D} := \mathcal{D} \cup \{(s, a, r + \gamma \max_{a' \in A} \hat{Q}_{s',a'})\}$

9 $\hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D})$

   - sample labels from out-dated $\hat{Q}$

2. relabel with current $\hat{Q}$:

8 $\mathcal{D} := \mathcal{D} \cup \{(s, a, r, s')\}$

9 $\mathcal{D}' := \{(s, a, r + \gamma \max_{a' \in A} \hat{Q}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}\}$

10 $\hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D}')$

   - sample labels correlate too heavily with current $\hat{Q}$ [Mnih et al., 2015]

Action Value Models Q / Relabeling (2/2)

3. relabel with mildly older $\hat{Q}$ (**target network**) [Mnih et al., 2015]:

8 $\quad \mathcal{D} := \mathcal{D} \cup \{(s, a, r, s')\}$

9 $\quad \mathcal{D}' := \{(s, a, r + \gamma \max_{a' \in A} \hat{Q}^{\text{target}}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}\}$

10 $\quad$ every $T_{\text{update-target}}$-th step:

11 $\quad\quad \hat{Q}^{\text{target}} := \hat{Q}$

12 $\quad \hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D}')$

Action Value Models Q / Relabeling (2/2)

3. relabel with mildly older $\hat{Q}$ (**target network**) [Mnih et al., 2015]:

8 $\mathcal{D} := \mathcal{D} \cup \{(s, a, r, s')\}$

9 $\mathcal{D}' := \{(s, a, r + \gamma \max_{a' \in A} \hat{Q}^{\text{target}}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}\}$

$= \{(s, a, r + \gamma \hat{Q}^{\text{target}}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}, a' := \arg\max_{a' \in A} \hat{Q}^{\text{target}}_{s',a'}\}$

10 every $T_{\text{update-target}}$-th step:

11 $\hat{Q}^{\text{target}} := \hat{Q}$

12 $\hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D}')$

- suffers from overestimation as same model is used to estimate the best action and its value [van Hasselt et al., 2016].

Action Value Models Q / Relabeling (2/2)

3. relabel with mildly older $\hat{Q}$ (**target network**) [Mnih et al., 2015]:

8 $\mathcal{D} := \mathcal{D} \cup \{(s, a, r, s')\}$

9 $\mathcal{D}' := \{(s, a, r + \gamma \max_{a' \in A} \hat{Q}^{\text{target}}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}\}$
$= \{(s, a, r + \gamma \hat{Q}^{\text{target}}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}, a' := \arg\max_{a' \in A} \hat{Q}^{\text{target}}_{s',a'}\}$

10 every $T_{\text{update-target}}$-th step:

11 $\hat{Q}^{\text{target}} := \hat{Q}$

12 $\hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D}')$

- suffers from overestimation as same model is used to estimate the best action and its value [van Hasselt et al., 2016].

4. relabel with mildly older $\hat{Q}$, but using action based on current $\hat{Q}$ (**double Q learning**) [van Hasselt et al., 2016]:

8 $\mathcal{D} := \mathcal{D} \cup \{(s, a, r, s')\}$

9 $\mathcal{D}' := \{(s, a, r + \gamma \hat{Q}^{\text{target}}_{s',a'}) \mid (s, a, r, s') \in \mathcal{D}, a' := \arg\max_{a' \in A} \hat{Q}_{s',a'}\}$

10 every $T_{\text{update-target}}$−th step:

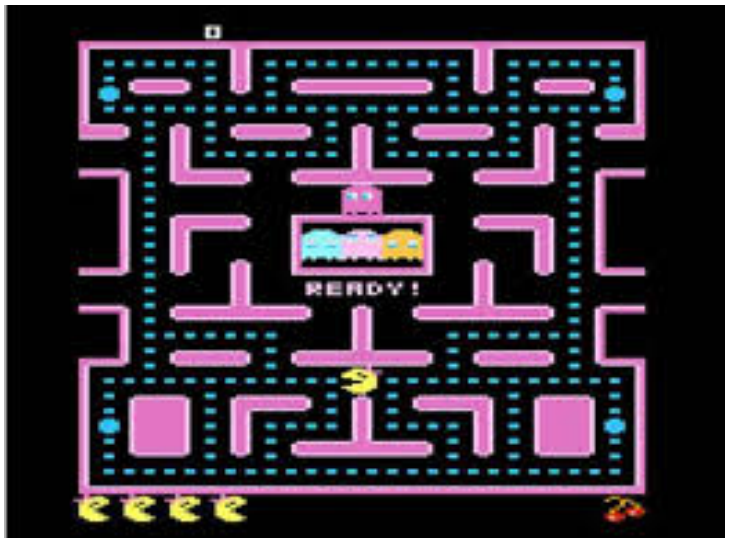
11 $\hat{Q}^{\text{target}} := \hat{Q}$

12 $\hat{Q} := \text{update-model}(\hat{Q}, \mathcal{D}')$

Outline

Deep Q Learning [Mnih et al., 2015]

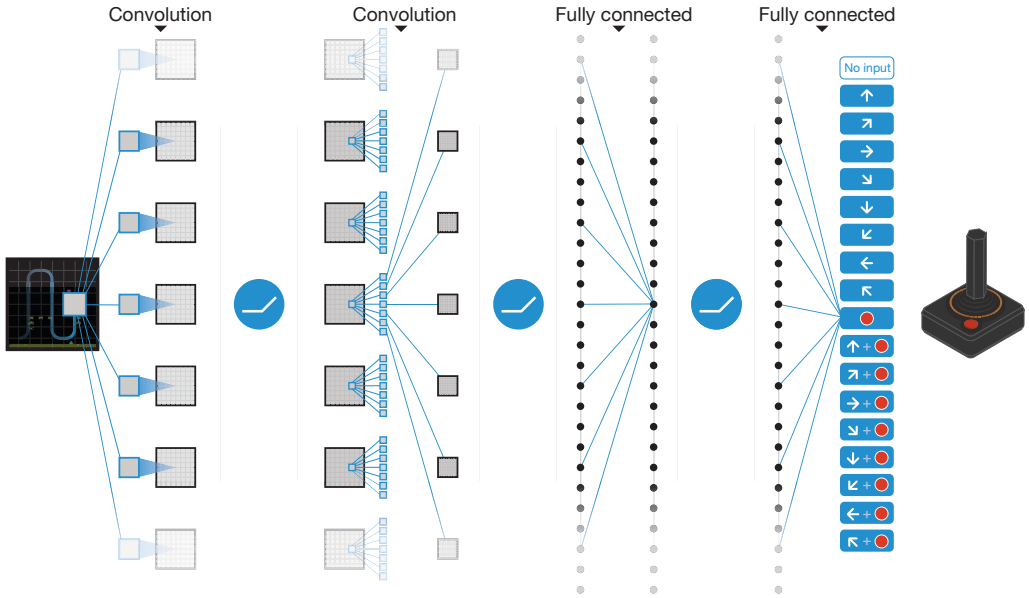
- task: learn to play atari games
  - input: last 4 images
  - output: 18 joystick movements
    - 9 directions times 2 (button pressed/not pressed)
- use a **deep convolutional neural network** for $\hat{Q}$



[source: https://ew.com/article/2013/01/25/the-10-best-atari-games/]

Deep Q Learning / Architecture

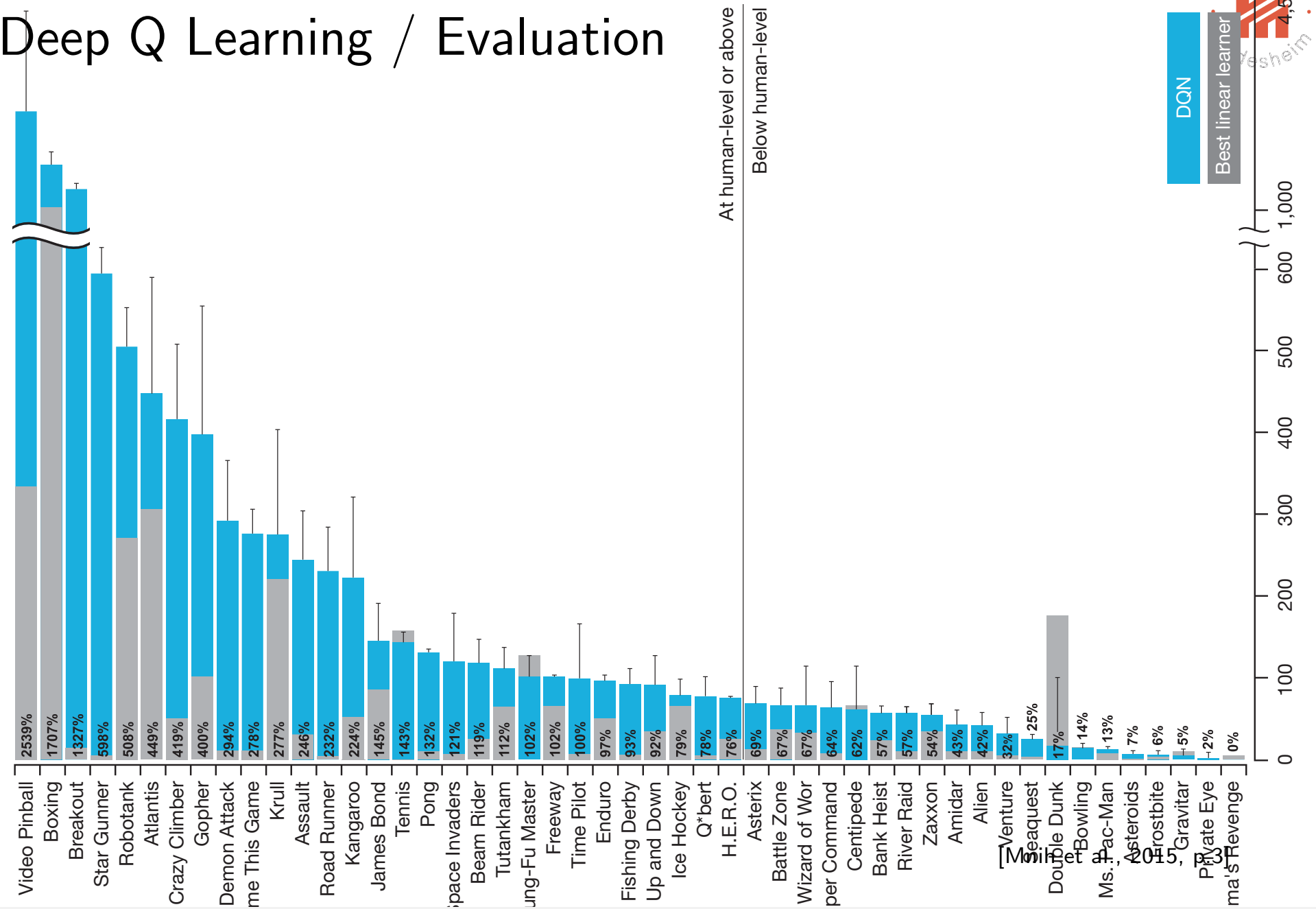
- state represented by 4 images of 84×84 pixels a 1 channel (luminance)



[source: [Mnih et al., 2015, p.2]]

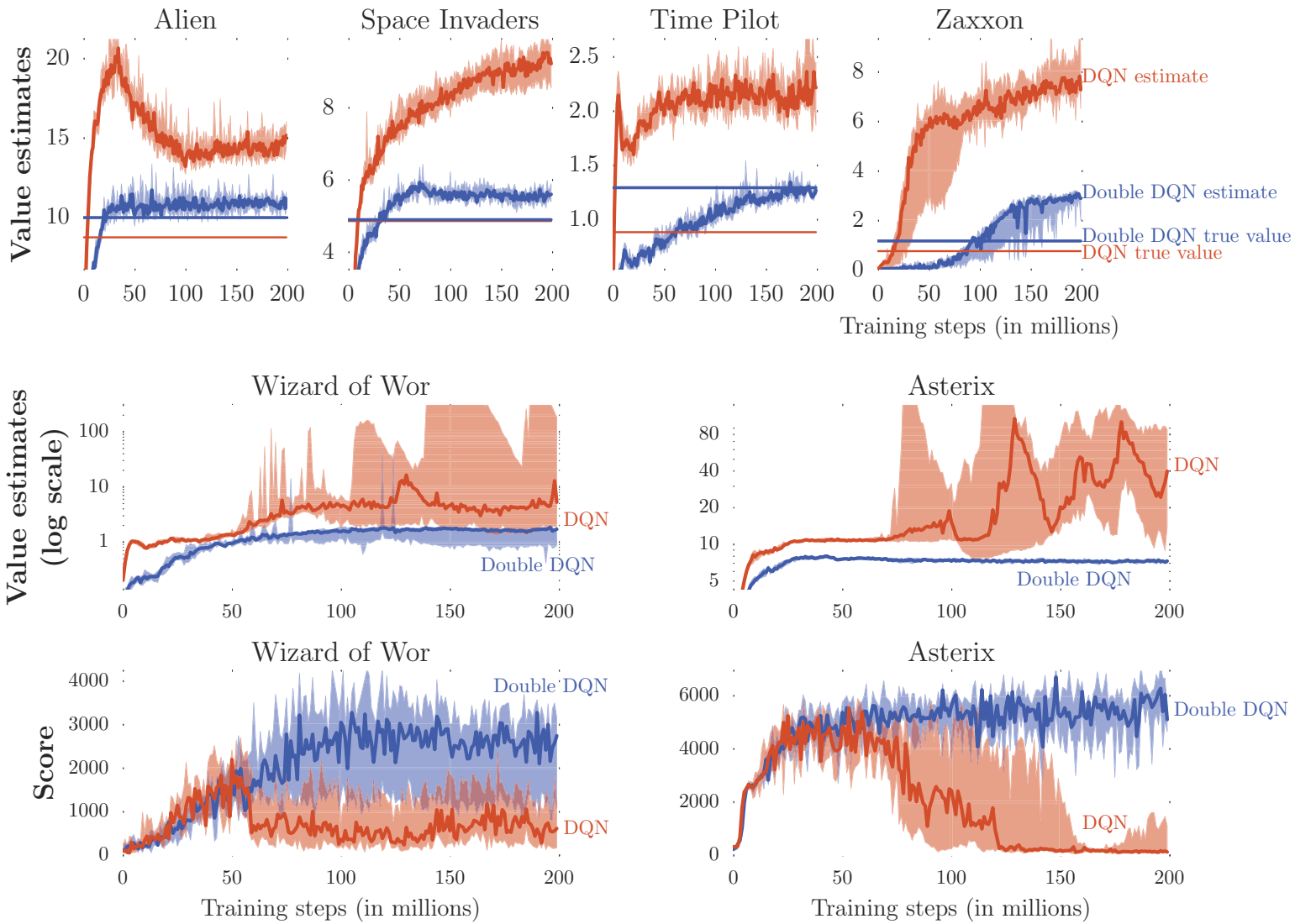
- layers:
  - input: $4 \times 84 \times 84$
  - conv. layer: 32 patterns 8×8 (stride 4), relu
  - conv. layer: 64 patterns 4×4 (stride 2), relu
  - conv. layer: 64 patterns 3×3 (stride 1), relu
  - fully connected: 512 nodes, relu
  - output: fully connected, 4–18 actions, softmax?

Deep Q Learning / Evaluation



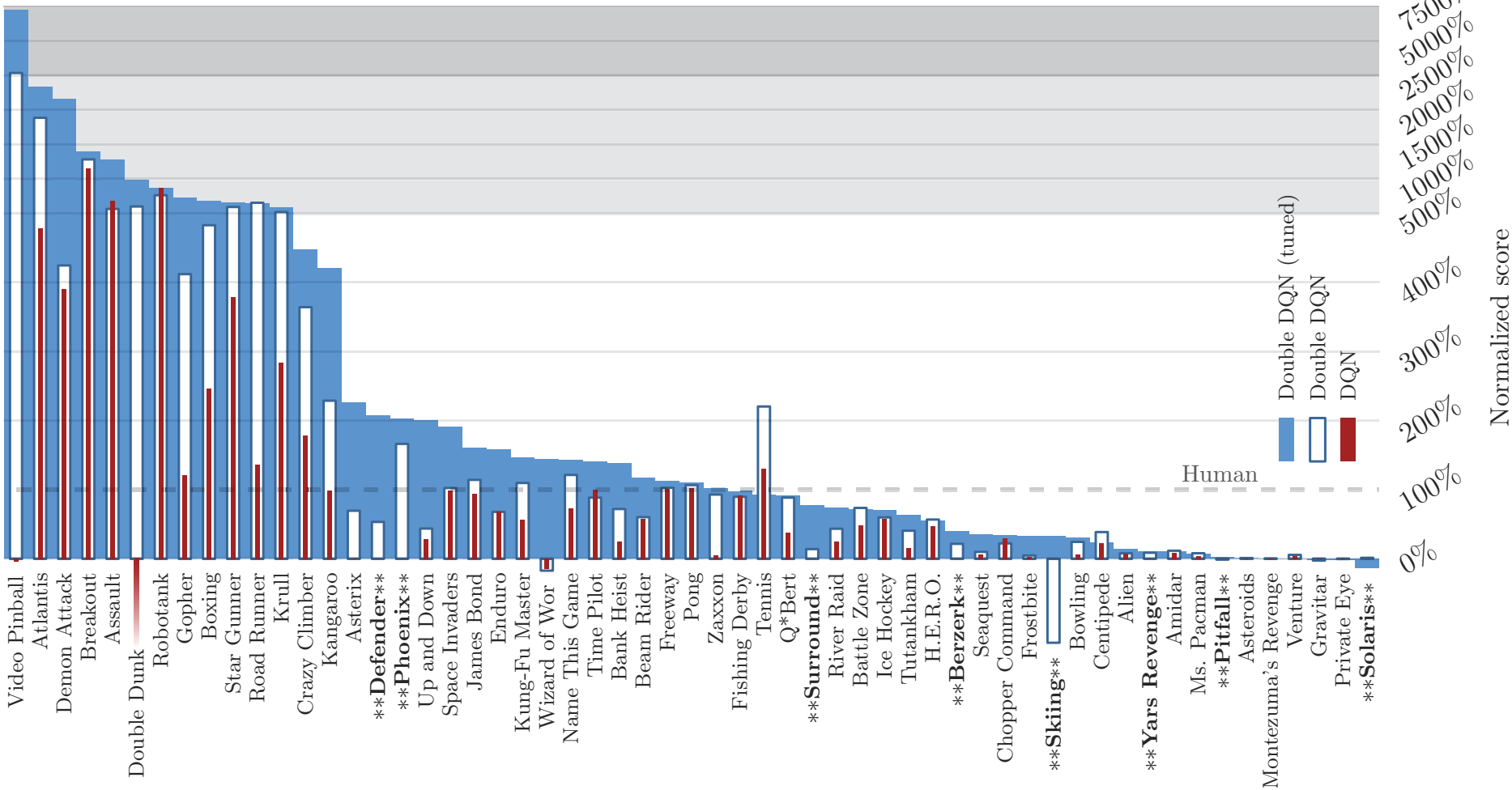
[Mnih et al., 2015, p. 3]

Deep Double Q Learning / Overestimation Effect



[van Hasselt et al., 2016, p.5]

Deep Double Q Learning / Evaluation



[van Hasselt et al., 2016, p.6]

Summary (1/2)

- **Q learning** learns the optimal action value function Q^* by updating Q estimates towards current reward plus estimated value of **best follow-up state**:

$$\hat{Q}_{s_t,a_t} = \hat{Q}_{s_t,a_t} + \alpha(r_t + \gamma \max_a \hat{Q}_{s_{t+1},a} - \hat{Q}_{s_t,a_t})$$

- For finite state and action spaces, tables can be used as models $\hat{Q} : S \times A \to \mathbb{R}$.
- Esp. for structured states (and actions), any regression model can be used.
  - e.g., deep neural networks (**deep Q learning**)
- The Q model should be learned also from past data, not just the current observation (**experience replay** vs. online learning).
  - Due to a non-stationary policy and the non-stationary labeling process, **old data should be discounted**.

Summary (2/2)

- Q Learning can be accellerated by relabeling past data
  - with a mildly older model (**target network**), or even better
  - with a mildly older model, choosing best actions by the current model (**double Q learning**).
- Deep reinforcement learning plays several video games **better than human players**.

Further Readings

- Reinforcement Learning:
  - Olivier Sigaud, Frederick Garica (2010): *Reinforcement Learning*, ch. 1 in Sigaud and Buffet [2010].
  - Sutton and Barto [2018]

References

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. ISSN 1476-4687. doi: 10.1038/nature14236.

Olivier Sigaud and Olivier Buffet, editors. *Markov Decision Processes in Artificial Intelligence*. Wiley, 2010.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, 2nd edition edition, 2018.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2094–2100, Phoenix, Arizona, 2016. AAAI Press.