<u>This lecture will be recorded!</u>

Welcome to

DS595 Reinforcement Learning

Prof. Yanhua Li

Time: 6:00pm –8:50pm W
Zoom Lecture
Fall 2022

Quiz 3 today Week 7 (3/2 W)

- Model free control (30 mins)

Quiz 4 in Week 9 (3/16 W)

- Linear Value Function Approximation（30 mins）
  - Stochastic Gradient Decent
  - VFA for policy evaluation
  - VFA for control

Project 3 is available
Due 3/23 Wed, Week #10
top three on the leader board get 10 bonus points

- https://users.wpi.edu/~yli15/courses/DS595Spring22/Assignments.html

- https://github.com/yingxue-zhang/DS595-RL-Projects/tree/master/Project3

Project 4 is available
Starts 3/23 Wed Week 10
Due 4/25 Monday Week 15

- https://users.wpi.edu/~yli15/courses/DS595Spring22/Assignments.html

- https://github.com/yingxue-zhang/DS595-RL-Projects/tree/master/Project4

A Project 4 self-intro session Wed in Week 9

We will have a

- Self Introduction Session on Wed in Week 9

- Who are you? Your expertise, such as programming experience, background knowledge of data mining, management, analytics.

- Experience on RL, Deep Learning, Data analytics

- Any initial idea for the open project 4?

Last lecture

- Non-linear value function approximation
  - Intro of Deep Reinforcement Learning (DRL)
  - Review on Deep Learning
  - Deep Q-Learning

This Lecture

- Advanced DQN methods
  - Double-DQN
  - Prioritized DQN
  - Dueling DQN

- Project 3 (by Yingxue) starting from around 8:20PM
  - Project 3 description
  - Pytorch configuration and Google cloud environment

	Reinforcement Learning	Inverse Reinforcement Learning
Single Agent	**Tabular representation of reward** Model-based control Model-free control (MC, SARSA, Q-Learning)	Linear reward function learning Imitation learning Apprenticeship learning Inverse reinforcement learning MaxEnt IRL MaxCausalEnt IRL MaxRelEnt IRL
	Function representation of reward *1. Linear value function approx* (MC, SARSA, Q-Learning) *2. Value function approximation* (Deep Q-Learning, Double DQN, prioritized DQN, Dueling DQN) *3. Policy function approximation* (Policy gradient, PPO, TRPO) *4. Actor-Critic methods* (A2C, A3C)	**Non-linear reward function learning** Generative adversarial imitation learning (GAIL) Adversarial inverse reinforcement learning (AIRL)
	Review of Deep Learning *As bases for non-linear function approximation (used in 2-4).*	**Review of Generative Adversarial nets**
Multiple Agents	**Multi-Agent Reinforcement Learning** Multi-agent Actor-Critic etc.	**Multi-Agent Inverse Reinforcement Learning** MA-GAIL MA-AIRL AMA-GAIL

Applications

This Lecture

- Advanced DQN methods
  - Double-DQN
  - Prioritized DQN
  - Dueling DQN

- Project 3 (by Yingxue) starting from around 8:20PM
  - Project 3 description
  - Pytorch configuration and Google cloud environment

Model-Free Deep Q-Learning

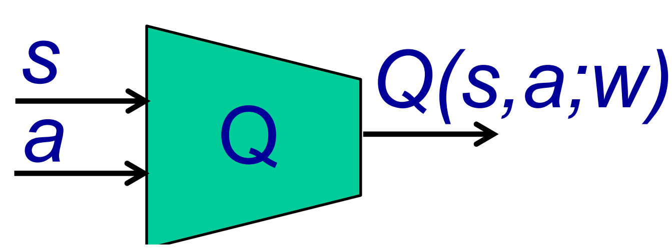
Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$
loop
  Sample tuple (s_k, a_k, r_k, s_{k+1}) given π
  Update weights:
$$\Delta w = -\alpha\left(r_k + \gamma \max_{a_{k+1}} \hat{Q}(s_{k+1}, a_{k+1}; w) - \hat{Q}(s_k, a_k; w)\right) \nabla_w \hat{Q}(s_k, a_k; w)$$
$$w = w - \Delta w$$
$$\pi(s_k) = \arg\max_{a_k} \hat{Q}(s_k, a_k), \text{with prob } 1 - \epsilon, \text{else random.}$$
  $k = k + 1$
end loop

\+ experience replay

reduce correlations between samples

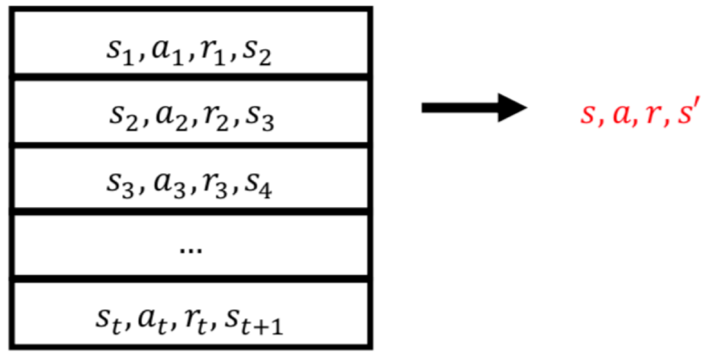
\+ fixed target

improve target stability



DQNs: Experience Replay

- To help remove correlations, store dataset (called a **replay buffer**) $\mathcal{D}$ from prior experience



- To perform experience replay, repeat the following:
  - $(s, a, r, s') \sim \mathcal{D}$: sample an experience tuple from the dataset
  - Compute the target value for the sampled s: $r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w})$
  - Use stochastic gradient descent to update the network weights

$$\Delta \mathbf{w} = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}))\nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

DQNs: Fixed Q-Targets

- To help improve stability, fix the **target weights** used in the target calculation for multiple updates
- Use a different set of weights to compute target than is being updated
- Let parameters $\boldsymbol{w}^-$ be the set of weights used in the target, and $\boldsymbol{w}$ be the weights that are being updated
- Slight change to computation of target value:
  - $(s, a, r, s') \sim \mathcal{D}$: sample an experience tuple from the dataset
  - Compute the target value for the sampled s: $r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}^-)$
  - Use stochastic gradient descent to update the network weights

$$\Delta \boldsymbol{w} = -\alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}^-) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$$

Periodically, update the fixed Q-target -network by the current Q-network.

DQN

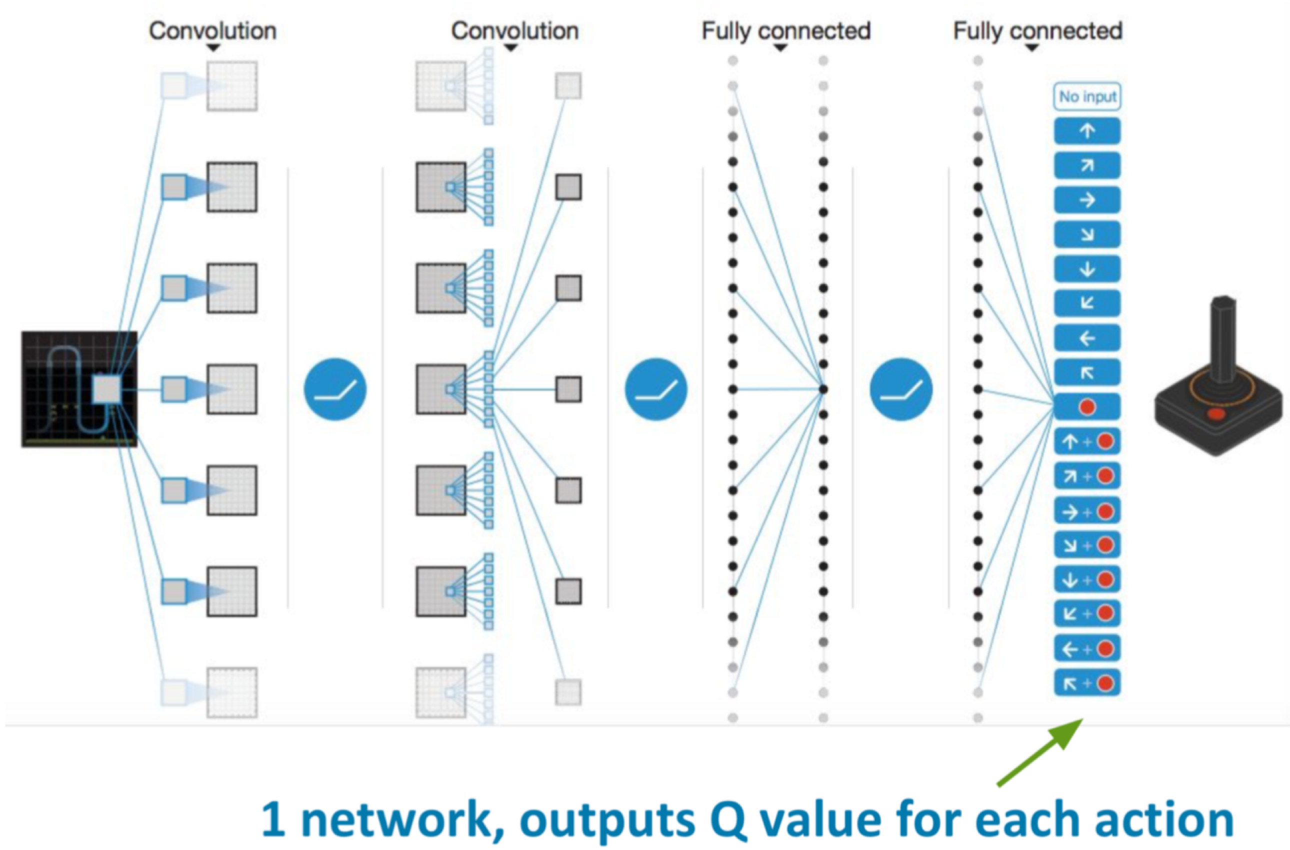


Figure: Human-level control through deep reinforcement learning, Mnih et al, 2015

Breakout game demo

https://www.youtube.com/watch?v=TmPfTpjtdgg

DQN Results in Atari

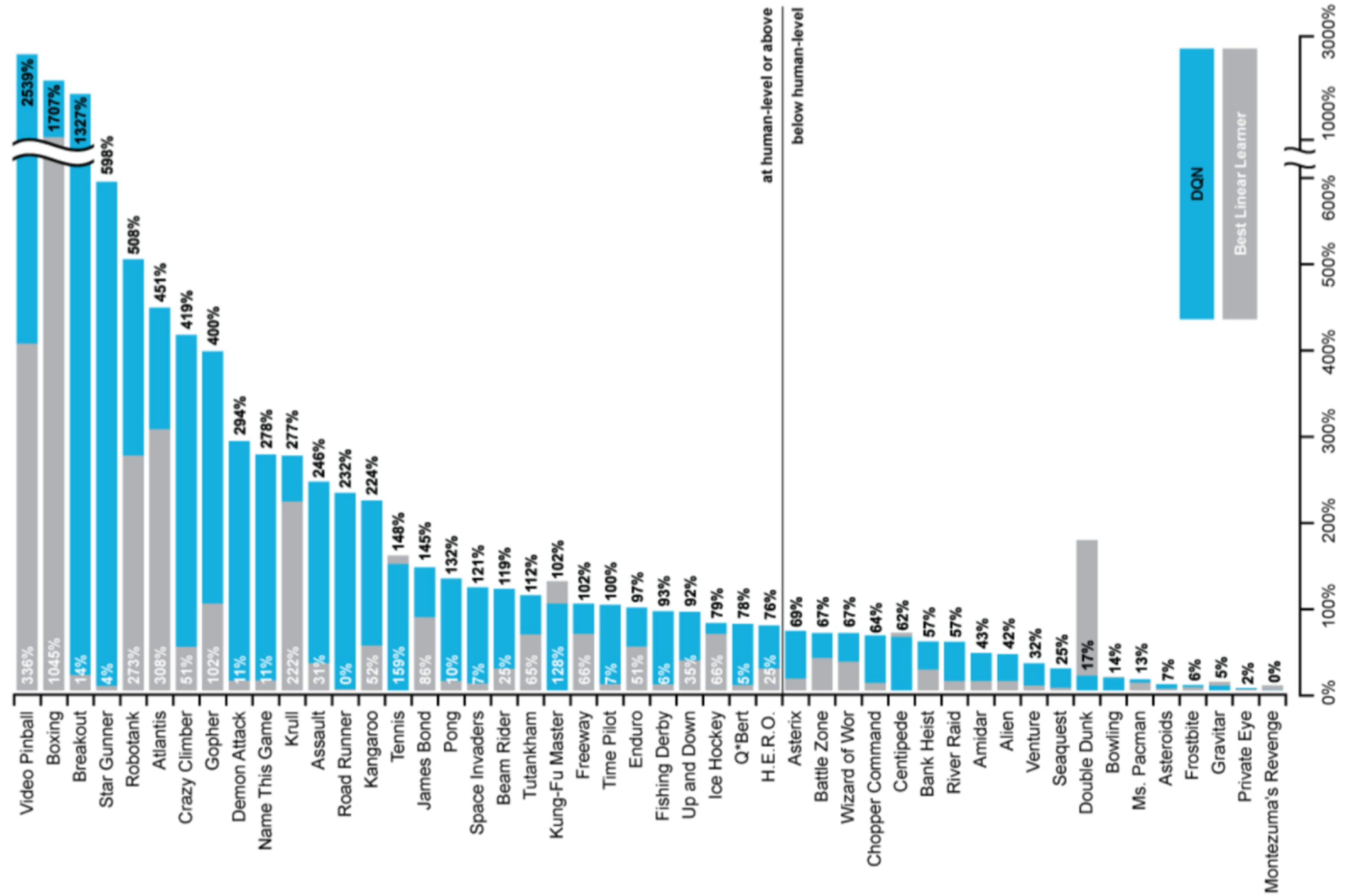


Figure: Human-level control through deep reinforcement learning, Mnih et al, 2015

Which Aspects of DQN were Important for Success?

Game	Linear	Deep Network	DQN w/ fixed Q	DQN w/ replay	DQN w/replay and fixed Q
Breakout	3	3	10	241	317
Enduro	62	29	141	831	1006
River Raid	2345	1453	2868	4102	7447
Seaquest	656	275	1003	823	2894
Space Invaders	301	302	373	826	1089

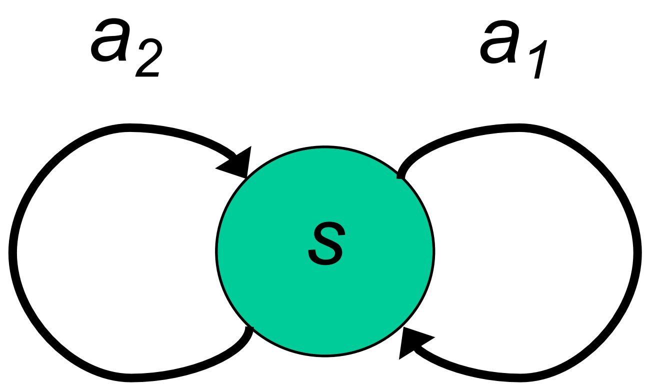
- Replay is **hugely** important

Deep RL

- Success in Atari has led to huge excitement in using deep neural networks to do value function approximation in RL
- Some immediate improvements (many others!)
  - **Double DQN** (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
  - Prioritized Replay (Prioritized Experience Replay, Schaul et al, ICLR 2016)
  - Dueling DQN (best paper ICML 2016) (Dueling Network Architectures for Deep Reinforcement Learning, Wang et al, ICML 2016)

A good link introducing all DQN's: https://medium.com/@parsa_h_m/deep-reinforcement-learning-dqn-double-dqn-dueling-dqn-noisy-dqn-and-dqn-with-prioritized-551f621a9823

Maximization bias

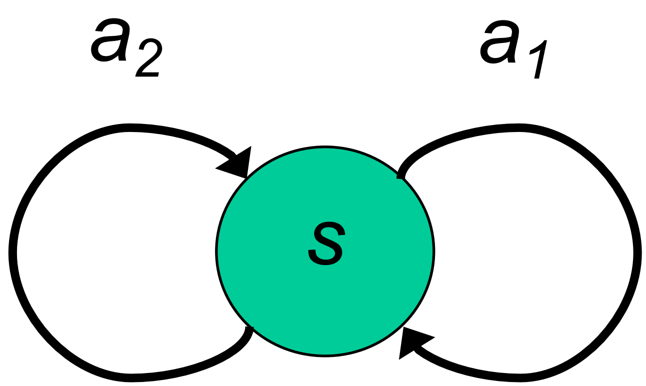


- $E(r|s, a = a_1) = E(r|s, a = a_2) = 0$
- Then $Q(s,a_1) = Q(s,a_2) = 0 = V(s)$ for any policy π
- Let $\hat{Q}(s, a_1), \hat{Q}(s, a_2)$ be the finite sample estimate of Q
- Use an unbiased estimator for Q: e.g. MC

$$\hat{Q}(s, a_1) = \frac{1}{n(s, a_1)} \sum_{i=1}^{n(s,a_1)} r_i(s, a_1)$$

- Let $\hat{\pi} = \arg\max_a \hat{Q}(s, a)$ be the greedy policy w.r.t. the estimated $\hat{Q}$. Even though each estimate of the state-action values $\hat{Q}$ is unbiased, the estimate of $\hat{\pi}$'s value $V^{\hat{\pi}}$ can be biased:

Maximization bias



- the estimate of $\hat{\pi}$'s value $V^{\hat{\pi}}$ can be biased:

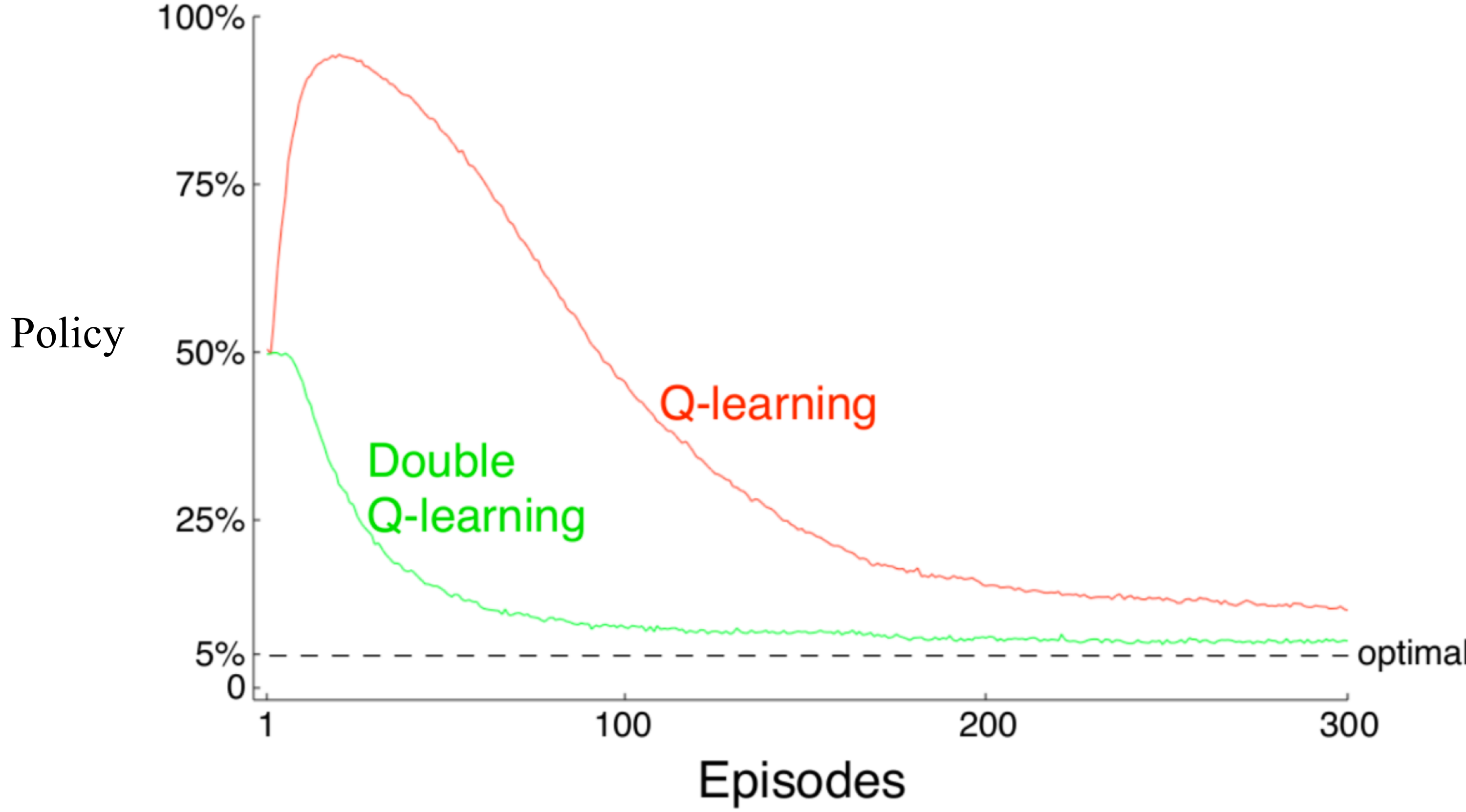
$$\hat{V}^{\hat{\pi}}(s) = \mathbb{E}[\max \hat{Q}(s, a_1), \hat{Q}(s, a_2)]$$
$$\geq \max[\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]]$$
$$= max[0, 0] = V^{\pi},$$

where the inequality comes from Jensen's inequality.

- Then the estimate of $\hat{Q}(s, a_1)$ will be over-estimated since $\hat{Q}(s, a_1) = r(s, a_1) + \boldsymbol{\gamma} V^{\hat{\pi}}(s')$

Double Q-Learning (Figure 6.7 in Sutton and Barto 2018)

[Double DQN] Deep Reinforcement Learning with Double Q-learning
Hado van Hasselt and Arthur Guez and David Silver Google DeepMind
https://arxiv.org/pdf/1509.06461.pdf



Due to the maximization bias, Q-learning spends much more time selecting suboptimal actions than double Q-learning.

Double Q-Learning

Recall: Version 1

Initialize $Q_1(s, a)$ and $Q_2(s, a)$,$\forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$

loop

  Select a_t using ϵ-greedy $\pi(s) = \arg\max_a Q_1(s_t, a) + Q_2(s_t, a)$

  Observe (r_t, s_{t+1})

  if (with 0.5 probability) **then** $a = \arg\max_a Q_1(s', a)$

    $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma * Q_2(s_{t+1}, a) - Q_1(s_t, a_t))$

  else $a = \arg\max_a Q_2(s', a)$

    $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + \gamma * Q_1(s_{t+1}, a) - Q_2(s_t, a_t))$

  end if

  $t = t + 1$

end loop

Separate <u>action selection</u> and <u>action evaluation</u>

---

https://papers.nips.cc/paper/3964-double-q-learning.pdf

- Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?
  Doubles the memory, same computation requirements, data requirements are subtle– might reduce amount of exploration needed due to lower bias

Double DQN

Separate action selection and action evaluation

- Extend this idea to DQN
- Current Q-network $\boldsymbol{w}$ is used to select actions
- Older Q-network $\boldsymbol{w}^-$ is used to evaluate actions

$$\Delta \boldsymbol{w} = \text{-}\alpha(r + \gamma \overbrace{\hat{Q}(\underbrace{\arg\max_{a'} \hat{Q}(s', a'; \boldsymbol{w})}_{\text{Action selection: } \boldsymbol{w}}; \boldsymbol{w}^-)}^{\text{Action evaluation: } \boldsymbol{w}^-} - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$$

In comparison to DQN below:

$$\Delta \boldsymbol{w} = \text{-}\alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}^-) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$$

Deep RL

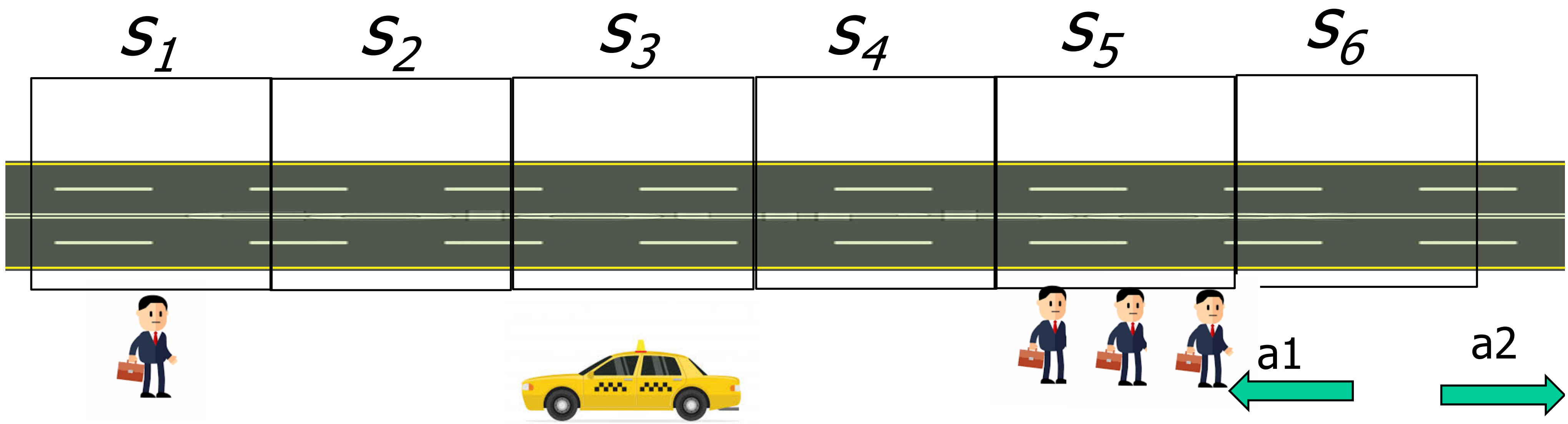
- Success in Atari has led to huge excitement in using deep neural networks to do value function approximation in RL
- Some immediate improvements (many others!)
  - DQN (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
  - **Prioritized Replay** (Prioritized Experience Replay, Schaul et al, ICLR 2016)
  - Dueling DQN (best paper ICML 2016) (Dueling Network Architectures for Deep Reinforcement Learning, Wang et al, ICML 2016)

Impact of Replay?

- In tabular TD-learning, **order** of replaying updates could help speed learning
- Repeating some updates seem to better propagate info than others
- Systematic ways to prioritize updates?

[ICLR 2016] PRIORITIZED EXPERIENCE REPLAY
Tom Schaul, John Quan, Ioannis Antonoglou and David Silver Google DeepMind
https://arxiv.org/pdf/1511.05952.pdf

Example: TD policy evaluation



Taxi passenger-seeking process: R=[1,0,0,0,3,0],

$\pi(s) = a_1$, $\forall s$, $\gamma = 1$. Any action from s1 and s6 terminates

Given (s_3, a_1, 0, s_3, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, T);

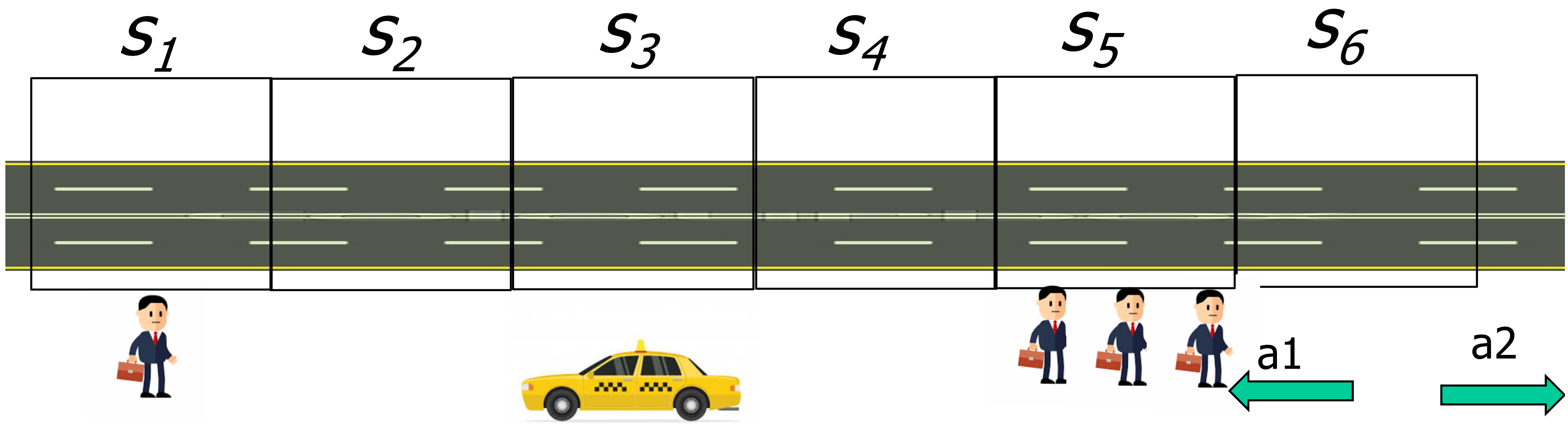
Q1: TD estimate of states (init at 0) with α=1? [100000]

Q2: Now get to choose 2 "replay" backups to do.

Which should we pick to get best estimate?

$$V^\pi(s) = V^\pi(s) + \alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s))$$

Example: TD policy evaluation



Taxi passenger-seeking process: R=[1,0,0,0,3,0]

For any action, π(s) = $a_{1,}$ ∀s, γ = 1.

any action from s1 and s6 terminates episode

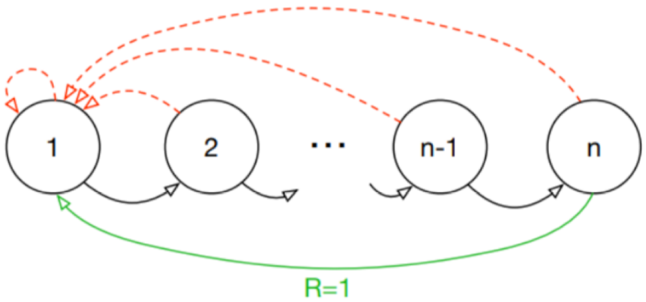
Given (s_3, a_1, 0, s_3, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, T);

Q1: TD estimate of states (init at 0) with α=1? [100000]

Q2: Now get to choose 2 "replay" backups to do. Which to pick to get best estimate? (s_2, a_1, 0, s_1), (s_3, a_1, 0, s_2)

$$V^\pi(s) = V^\pi(s) + \alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s))$$

Potential Impact of Ordering Episodic Replay Updates



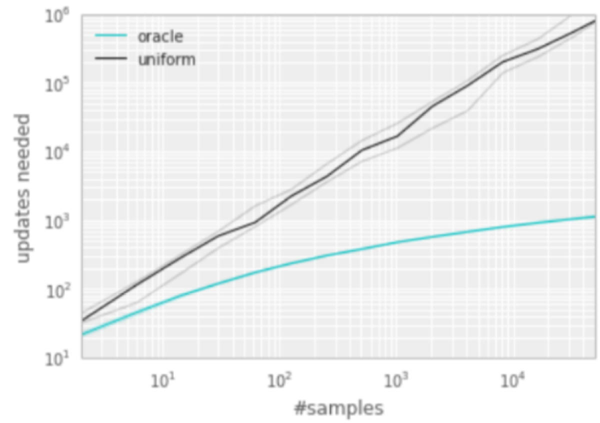


Figure: Schaul, Quan, Antonoglou, Silver ICLR 2016

- Schaul, Quan, Antonoglou, Silver ICLR 2016
- Oracle: picks (s, a, r, s') tuple to replay that will minimize global loss
- Exponential improvement in convergence
  - Number of updates needed to converge
- Oracle is not a practical method but illustrates impact of ordering

Prioritized Experience Replay

- Let i be the index of the i-the tuple of experience (s_i, a_i, r_i, s_{i+1})
- Sample tuples for update using priority function
- Priority of a tuple i is proportional to DQN error

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \mathbf{w}^-) - Q(s_i, a_i; \mathbf{w}) \right|$$

- Update p_i every update
- p_i for new tuples is set to 0
- One method[1]: proportional (stochastic prioritization)

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

Check Your Understanding

- Let i be the index of the i-the tuple of experience (s_i, a_i, r_i, s_{i+1})
- Sample tuples for update using priority function
- Priority of a tuple i is proportional to DQN error

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

- Update p_i every update
- p_i for new tuples is set to 0
- One method[1]: proportional (stochastic prioritization)

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

- $\alpha = 0$ yields what rule for selecting among existing tuples?

[1] See paper for details and an alternative

Deep RL

- Success in Atari has led to huge excitement in using deep neural networks to do value function approximation in RL
- Some immediate improvements (many others!)
  - DQN (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
  - Prioritized Replay (Prioritized Experience Replay, Schaul et al, ICLR 2016)
  - **Dueling DQN** (best paper ICML 2016) (Dueling Network Architectures for Deep Reinforcement Learning, Wang et al, ICML 2016)

Value & Advantage Function

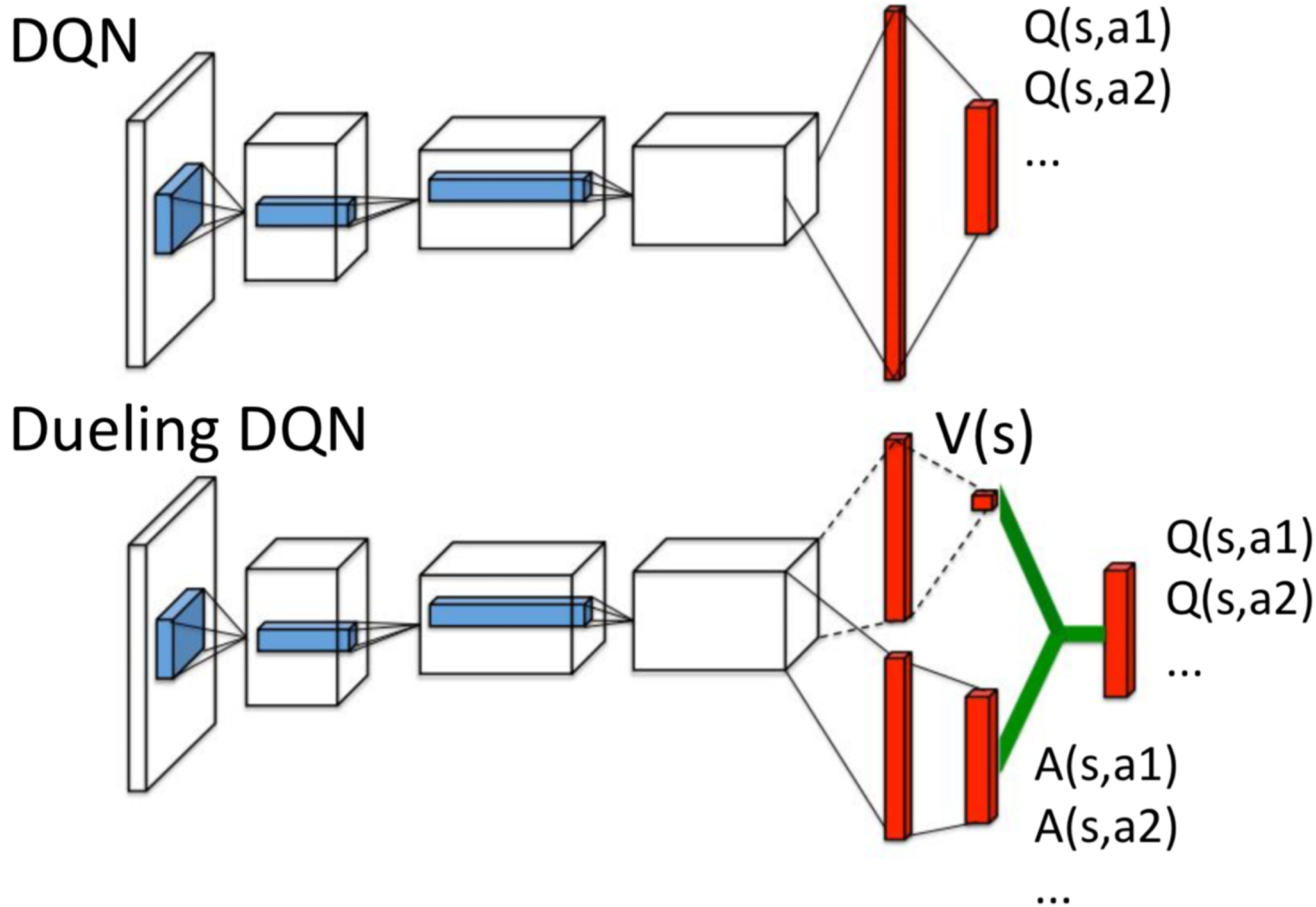
- Intuition: Features need to pay attention to determine value may be different than those need to determine action benefit
- E.g.
  - Game score may be relevant to predicting $V(s)$
  - But not necessarily in indicating relative action values
- Advantage function (Baird 1993)

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

overtime the Q-value would not overshoot, with $E_{a \sim \pi}(s)\ [A^{\pi}\ (s, a)] = 0$

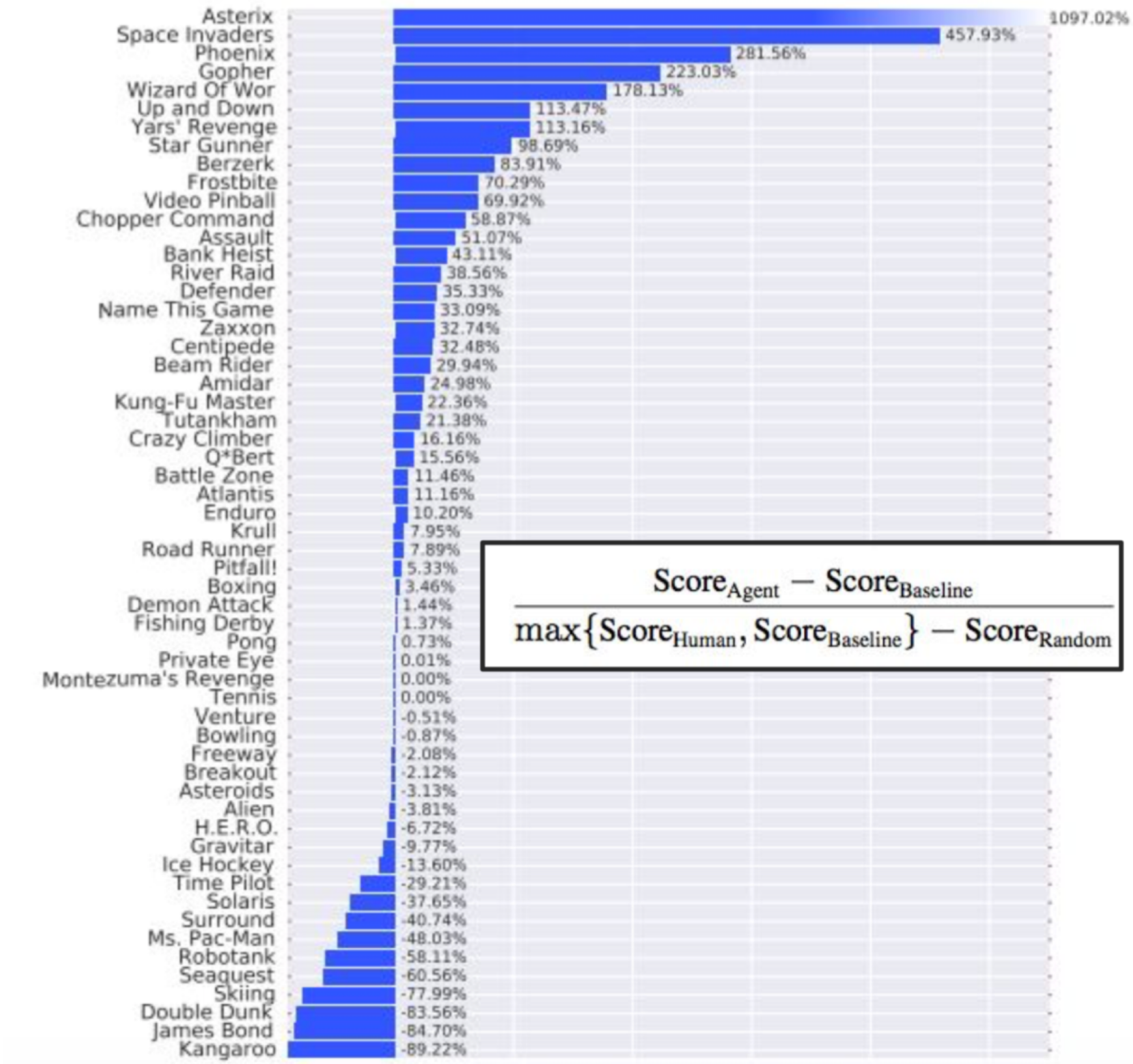
[ICML 2016] Dueling Network Architectures for Deep Reinforcement Learning
Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas
https://arxiv.org/pdf/1511.06581.pdf

Dueling DQN



Wang et.al., ICML, 2016

Dueling DQN V.S. Double DQN with Prioritized Replay

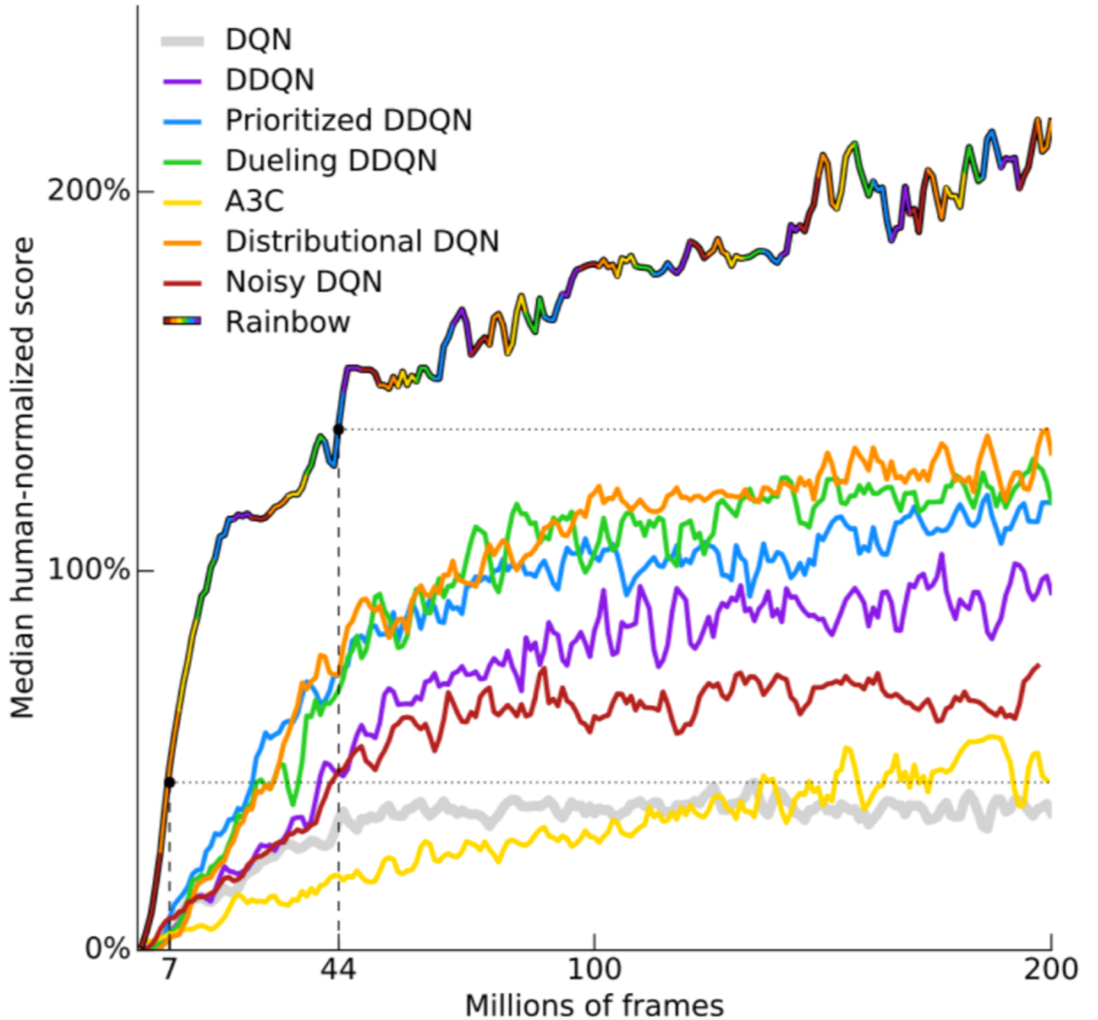


$$\frac{\text{Score}_{\text{Agent}} - \text{Score}_{\text{Baseline}}}{\max\{\text{Score}_{\text{Human}}, \text{Score}_{\text{Baseline}}\} - \text{Score}_{\text{Random}}}$$

Comparison with Double DQN (as the baseline)

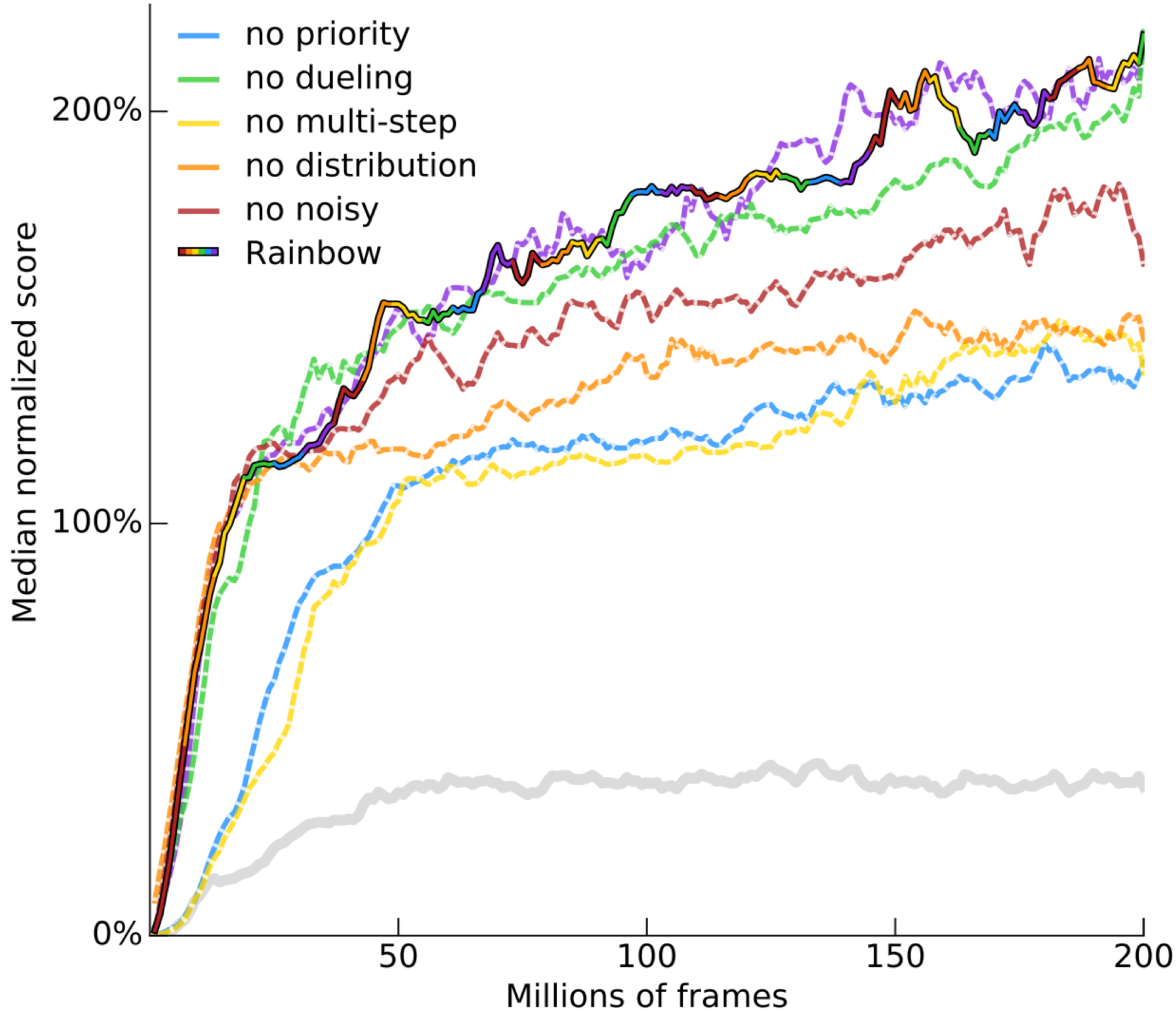
Figure: Wang et al, ICML 2016

Deep Reinforcement Learning



[AAAI 2018] Rainbow: Combining Improvements in Deep Reinforcement Learning,
Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney,
Dan Horgan, Bilal Piot, Mohammad Azar, David Silver, AAAI 2018,
https://arxiv.org/pdf/1710.02298.pdf

Deep Reinforcement Learning



[AAAI 2018] Rainbow: Combining Improvements in Deep Reinforcement Learning, Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, David Silver, AAAI 2018, https://arxiv.org/pdf/1710.02298.pdf

Next Lecture

- Other deep reinforcement learning approaches

  - Value based DRL (DQN),

  - Policy based DRL
    - Policy Gradient
    - Proximal Policy Optimization, PPO, -> PPO2
    - TRPO (Trust Region Policy Optimization, TRPO

  - (Asynchronous) Advantage Actor Critic:
    - A2C
    - A3C

	Reinforcement Learning	Inverse Reinforcement Learning
Single Agent	**Tabular representation of reward** Model-based control Model-free control (MC, SARSA, Q-Learning)	Linear reward function learning Imitation learning Apprenticeship learning Inverse reinforcement learning MaxEnt IRL MaxCausalEnt IRL MaxRelEnt IRL
	Function representation of reward *1. Linear value function approx* (MC, SARSA, Q-Learning) *2. Value function approximation* (Deep Q-Learning, Double DQN, prioritized DQN, Dueling DQN) *3. Policy function approximation* (Policy gradient, PPO, TRPO) *4. Actor-Critic methods* (A2C, A3C)	**Non-linear reward function learning** Generative adversarial imitation learning (GAIL) Adversarial inverse reinforcement learning (AIRL)
	Review of Deep Learning *As bases for non-linear function approximation (used in 2-4).*	**Review of Generative Adversarial nets**
Multiple Agents	**Multi-Agent Reinforcement Learning** Multi-agent Actor-Critic etc.	**Multi-Agent Inverse Reinforcement Learning** MA-GAIL MA-AIRL AMA-GAIL

Applications

Questions?