

Lecture 7: Deep RL Continued

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2022

Refresh Your Knowledge LN7

- Experience replay in deep Q-learning (select all):
 - 1 Involves using a bank of prior (s,a,r,s') tuples and doing Q-learning updates on the tuples in the bank
 - 2 Always uses the most recent history of tuples
 - 3 Reduces the data efficiency of DQN
 - 4 Increases the computational cost
 - 5 Not sure

Refresh Your Knowledge 6 Solutions

- Experience replay in deep Q-learning (select all):
 - ① Involves using a bank of prior (s,a,r,s') tuples and doing Q-learning updates on the tuples in the bank
 - ② Always uses the most recent history of tuples
 - ③ Reduces the data efficiency of DQN
 - ④ Increases the computational cost
 - ⑤ Not sure

Answer: It increases the computational cost, it uses a bank of tuples and it samples them, it's likely to **improve** the data efficiency, and it does not have to always use the most recent history of tuples.

Class Structure

- Last time: CNNs and Deep Reinforcement learning
- This time: Deep RL
- Next time: Policy Search

- 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)

- 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)
- Practical Tips

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

- Recall maximization bias challenge
 - Max of the estimated state-action values can be a biased estimate of the max
- Double Q-learning

Recall: Double Q-Learning

```
1: Initialize  $Q_1(s, a)$  and  $Q_2(s, a), \forall s \in S, a \in A$   $t = 0$ , initial state  $s_t = s_0$ 
2: loop
3:   Select  $a_t$  using  $\epsilon$ -greedy  $\pi(s) = \arg \max_a Q_1(s_t, a) + Q_2(s_t, a)$ 
4:   Observe  $(r_t, s_{t+1})$ 
5:   if (with 0.5 probability) then
6:
       
$$Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + Q_2(s_{t+1}, \arg \max_{a'} Q_1(s_{t+1}, a')) - Q_1(s_t, a_t))$$

7:
       else
8:
       
$$Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + Q_1(s_{t+1}, \arg \max_{a'} Q_2(s_{t+1}, a')) - Q_2(s_t, a_t))$$

9:   end if
10:   $t = t + 1$ 
11: end loop
```

Double DQN

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

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

Action evaluation: \mathbf{w}^-

does not eliminate max bias

Double DQN

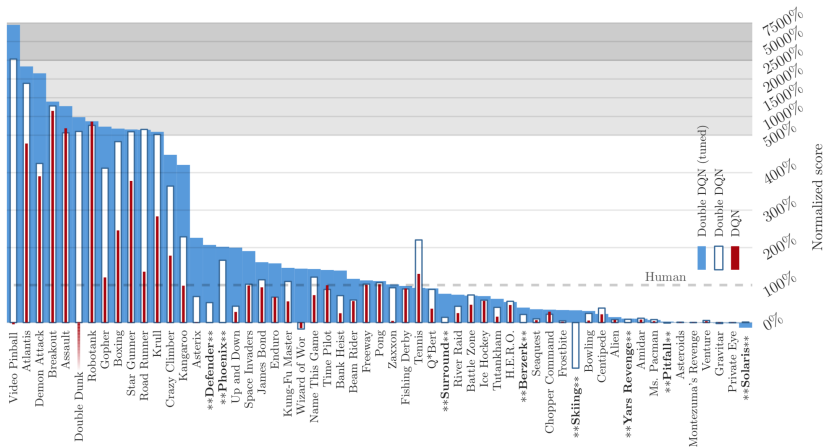


Figure: van Hasselt, Guez, Silver, 2015

- 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)
- Practical Tips

Check Your Understanding: Mars Rover Model-Free Policy Evaluation

s_1	s_2	s_3	s_4	s_5	s_6	s_7
$R(s_1) = +1$ <i>Okay</i> <i>Field Site</i>	$R(s_2) = 0$	$R(s_3) = 0$	$R(s_4) = 0$	$R(s_5) = 0$	$R(s_6) = 0$	$R(s_7) = +10$ <i>Fantastic</i> <i>Field Site</i>

- $\pi(s) = a_1 \forall s, \gamma = 1$. Any action from s_1 and s_7 terminates episode
- Trajectory = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, \text{terminal})$
- First visit MC estimate of V of each state? $[1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$
- TD estimate of all states (init at 0) with $\alpha = 1$ is $[1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
- Choose 2 additional "replay" backups to do. Which should we pick to get a V estimate closest to MC first visit estimate?
 - 1 Doesn't matter, any will yield the same
 - 2 $(s_3, a_1, 0, s_2)$ then $(s_2, a_1, 0, s_1)$
 - 3 $(s_2, a_1, 0, s_1)$ then $(s_2, a_1, 0, s_2)$
 - 4 $(s_2, a_1, 0, s_1)$ then $(s_3, a_1, 0, s_2)$
 - 5 Not sure

Check Your Understanding: Mars Rover Model-Free Policy Evaluation Solution

s_1	s_2	s_3	s_4	s_5	s_6	s_7
$R(s_1) = +1$ <i>Okay Field Site</i>	$R(s_2) = 0$	$R(s_3) = 0$	$R(s_4) = 0$	$R(s_5) = 0$	$R(s_6) = 0$	$R(s_7) = +10$ <i>Fantastic Field Site</i>

- $\pi(s) = a_1 \forall s, \gamma = 1$. Any action from s_1 and s_7 terminates episode
- Trajectory = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, \text{terminal})$
- First visit MC estimate of V of each state? $[1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$
- TD estimate of all states (init at 0) with $\alpha = 1$ is $[1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
- Choose 2 additional "replay" backups to do. Which should we pick to get a V estimate closest to MC first visit estimate?
 - 1 Doesn't matter, any will yield the same
 - 2 $(s_3, a_1, 0, s_2)$ then $(s_2, a_1, 0, s_1)$
 - 3 $(s_2, a_1, 0, s_1)$ then $(s_2, a_1, 0, s_2)$
 - 4 $(s_2, a_1, 0, s_1)$ then $(s_3, a_1, 0, s_2)$
 - 5 Not sure

Answer: $(s_2, a_1, 0, s_1), (s_3, a_1, 0, s_2)$ yielding $V = [1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$.

$$V = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]$$

Impact of Replay?

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

Potential Impact of Ordering Episodic Replay Updates

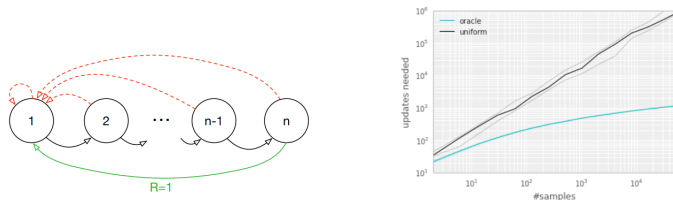


Figure: Schaul, Quan, Antonoglou, Silver ICLR 2016

- 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 -th 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 maximum value
- One method¹: proportional (stochastic prioritization)

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

parameter

¹See paper for details and an alternative

Check Your Understanding: Prioritized Replay

- Let i be the index of the i -th 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.
- One method [See paper for details]: proportional (stochastic prioritization)

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

- $\beta = 0$ yields what rule for selecting among existing tuples?
- Selects randomly
- Selects the one with the highest priority
- It depends on the priorities p of the tuples
- Not Sure

Check Your Understanding: Prioritized Replay

- Let i be the index of the i -th 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.
- One method¹: proportional (stochastic prioritization)

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

- $\beta = 0$ yields what rule for selecting among existing tuples?
- Selects randomly
- Selects the one with the highest priority
- It depends on the priorities of the tuples
- Not Sure

Answer: Selects randomly

Performance of Prioritized Replay vs Double DQN

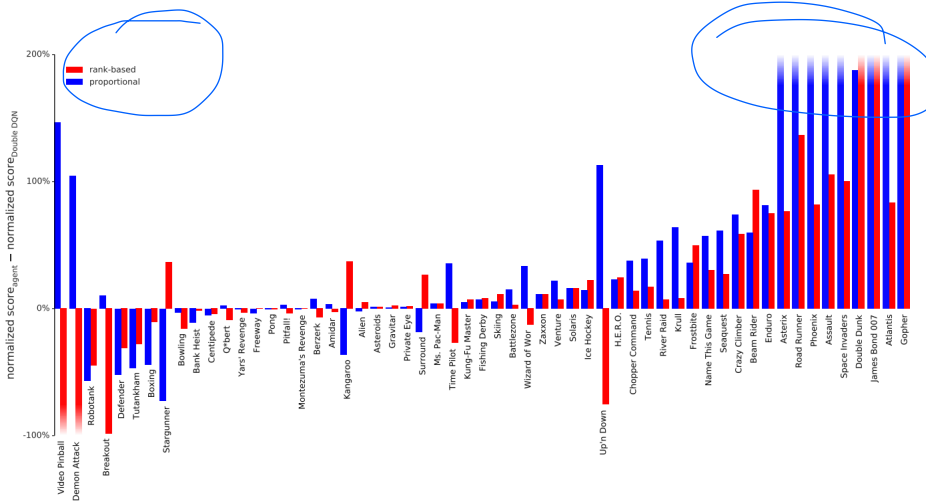


Figure: Schaul, Quan, Antonoglou, Silver ICLR 2016

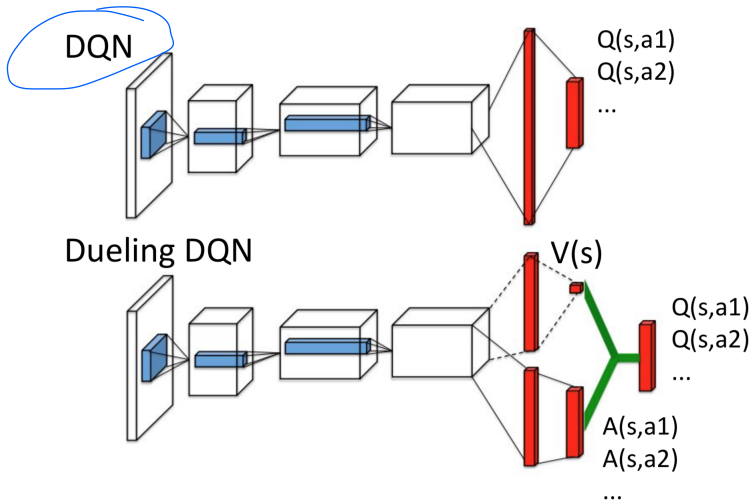
- Double DQN (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
- Prioritized Replay (Prioritized Experience Replay, Schaul et al, ICLR 2016)
- Dueling DQN (best paper ICML 2016) (Dueling Network Architectures for Deep Reinforcement Learning, Wang et al, ICML 2016)
- Practical Tips

Value & Advantage Function

- Intuition: Features needed to accurately represent value may be different than those needed to specify difference in actions
- E.g.
 - Game score may help accurately predict $V(s)$
 - But not necessarily in indicating relative action values $Q(s, a_1)$ vs $Q(s, a_2)$
- Advantage function (Baird 1993)

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

Dueling DQN



Wang et.al., ICML, 2016

Advantage Function and Training

- Advantage function

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

- Consider a network that outputs $V(s; \theta, \beta)$ as well as advantage $A(s, a; \theta, \lambda)$ where θ, β , and λ are parameters
- To construct Q could use $Q(s, a; \theta, \beta, \lambda) = V(s; \theta, \beta) + A(s, a; \theta, \lambda)$
- Do we expect that this architecture will result in learning a good estimate of true V or A ?

Check Your Understanding: Unique?

- Advantage function

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

- For a given Q function, is there a unique A advantage function and V ?
 - 1 Yes
 - 2 No
 - 3 Not sure

Check Your Understanding: Unique?

- Advantage function

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

- For a given Q function, is there a unique A advantage function and V ?
 - 1 Yes
 - 2 No
 - 3 Not sure

Answer: No. If we are just given a Q , there are many A and V that could satisfy this – for example, by shifting things by a constant. This can cause challenges for using the simple proposal before:

$$Q(s, a; \theta, \beta, \lambda) = V(s; \theta, \beta) + A(s, a; \theta, \lambda)$$

Uniqueness

- Consider a network that outputs $V(s; \theta, \beta)$ as well as advantage $A(s, a; \theta, \lambda)$ where θ, β , and λ are parameters
- To construct Q could use $Q(s, a; \theta, \beta, \lambda) = V(s; \theta, \beta) + A(s, a; \theta, \lambda)$
- Option 1: Force $Q(s, a) = V(s)$ for the best action suggested by the advantage:

$$\hat{Q}(s, a; \mathbf{w}) = \hat{V}(s; \mathbf{w}) + \left(\hat{A}(s, a; \mathbf{w}) - \max_{a' \in \mathcal{A}} \hat{A}(s, a'; \mathbf{w}) \right)$$

- This helps force the V network to approximate V

Uniqueness

- Consider a network that outputs $V(s; \theta, \beta)$ as well as advantage $A(s, a; \theta, \lambda)$ where θ, β , and λ are parameters
- To construct Q could use $Q(s, a; \theta, \beta, \lambda) = V(s; \theta, \beta) + A(s, a; \theta, \lambda)$
- Option 1: Force $Q(s, a) = V(s)$ for the best action suggested by the advantage:

$$\hat{Q}(s, a; \mathbf{w}) = \hat{V}(s; \mathbf{w}) + \left(\hat{A}(s, a; \mathbf{w}) - \max_{a' \in \mathcal{A}} \hat{A}(s, a'; \mathbf{w}) \right)$$

- This helps force the V network to approximate V
- Option 2: Use mean as baseline (more stable)

$$\hat{Q}(s, a; \mathbf{w}) = \hat{V}(s; \mathbf{w}) + \left(\hat{A}(s, a; \mathbf{w}) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} \hat{A}(s, a'; \mathbf{w}) \right)$$

- More stable often because averaging over all advantages instead of the advantage of the current max action.

Dueling DQN V.S. Double DQN with Prioritized Replay

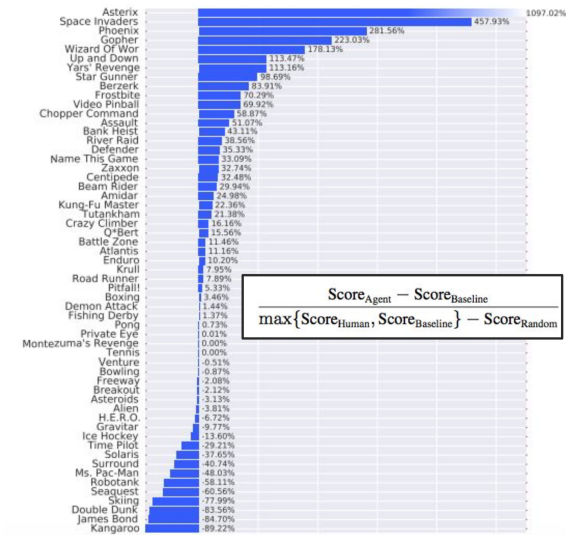


Figure: Wang et al, ICML 2016

- 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)
- **Practical Tips**

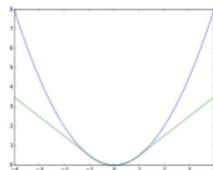
Practical Tips for DQN on Atari (from J. Schulman)

- DQN is more reliable on some Atari tasks than others. Pong is a reliable task: if it doesn't achieve good scores, something is wrong
- Large replay buffers improve robustness of DQN, and memory efficiency is key
 - Use uint8 images, don't duplicate data
- Be patient. DQN converges slowly—for ATARI it's often necessary to wait for 10-40M frames (couple of hours to a day of training on GPU) to see results significantly better than random policy
- In our Stanford class: Debug implementation on small test environment

Practical Tips for DQN on Atari (from J. Schulman) cont.

- Try Huber loss on Bellman error

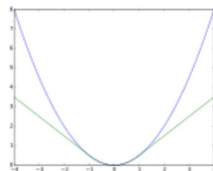
$$L(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq \delta \\ \delta|x| - \frac{\delta^2}{2} & \text{otherwise} \end{cases}$$



Practical Tips for DQN on Atari (from J. Schulman) cont.

- Try Huber loss on Bellman error

$$L(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq \delta \\ \delta|x| - \frac{\delta^2}{2} & \text{otherwise} \end{cases}$$

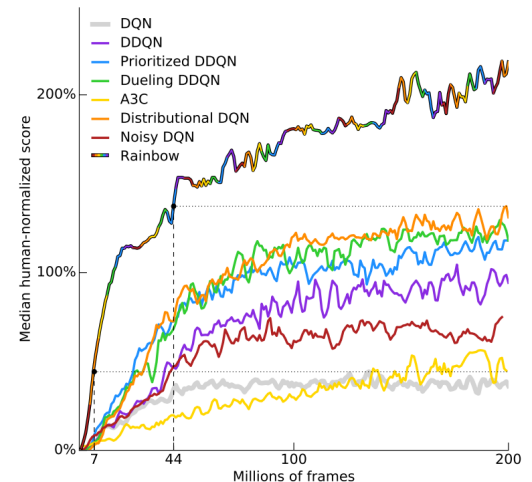


- Consider trying Double DQN—significant improvement from small code change
- To test out your data pre-processing, try your own skills at navigating the environment based on processed frames 10 - 20
- Always run at least two different seeds when experimenting
- Learning rate scheduling is beneficial. Try high learning rates in initial exploration period
- Try non-standard exploration schedules

Recap: Deep Model-free RL, 3 of the Early Big Ideas

- 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)

Deep Reinforcement Learning 2018



- Hessel, Matteo, et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning."
- Very active area of research!

Summary of Model Free Value Function Approximation with DNN & What You Should Know

- DNN are very expressive function approximators
- Can use DNNs to represent the Q function and do MC or TD style methods
- You should be able to implement DQN (assignment 2)
- You should be able to list a few extensions that help performance beyond DQN

Class Structure

- Last time: CNNs and Deep Reinforcement learning
- This time: Deep RL
- Next time: Policy Search