

# Lecture 16 - Deep Q-learning

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning  
Course Page: [\[Click\]](#)

## Recap: Q-learning in discrete state space

Recall that Q-learning is an off-policy method for TD-style control. Despite that it is off-policy, we do not need to rely on importance sampling. Instead, we can maintain the Q estimates and bootstrap the value of the best future action.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left( r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right).$$

As we take a maximum over the actions at the next state, this action is not necessarily the same as the one we would derive from the current policy. Therefore, Q-learning is considered an off-policy algorithm.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

## Recap: Q-learning in discrete state space

---

### Algorithm 5: Q-learning with $\epsilon$ -greedy exploration

---

**Input:**  $\epsilon, \alpha, \gamma$

Initialize  $Q(s, a)$  for all  $s \in S, a \in A$  arbitrarily except  $Q(\text{terminal}, \cdot) = 0$

$\pi \leftarrow \epsilon$ -greedy policy with respect to  $Q$

**for each episode do**

$t \leftarrow 1$

    Set  $s_1$  as the starting state

**while** *until episode terminates do*

        Sample action  $a_t$  from policy  $\pi(s_t)$

        Take action  $a_t$  and observe reward  $r_t$  and next state  $s_{t+1}$

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

$\pi \leftarrow \epsilon$ -greedy policy with respect to  $Q$

$t \leftarrow t + 1$

**return**  $Q, \pi$

---

# Value-based deep reinforcement learning

In this lecture, we introduce three popular value-based deep reinforcement learning (RL) algorithms: **Deep Q-network (DQN)**, **Double DQN** and **Dueling DQN**.

- They can learn successful policies directly from **high-dimensional inputs**, (e.g. pre-processed pixels from video games) by using end-to-end reinforcement learning.
- They achieved a level of performance that is comparable to a **professional human games** tester across a set of 49 names on Atari 2600.
- **Convolutional neural networks (ConvNets)** are used in these architectures for feature extraction from pixel inputs, as an example of feature representations.

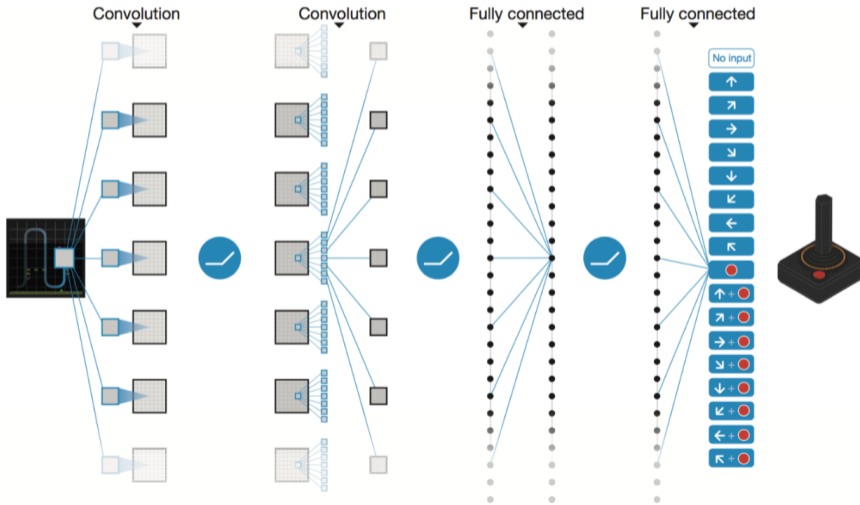


香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Generalization: Deep Q-network (DQN)

DQN architecture: neural networks are used as function approximations.



(深圳)  
f Hong Kong, Shenzhen

# Generalization: Deep Q-network (DQN)

**DQN architecture:** neural networks are used as function approximations.

- The network takes pre-processed **pixel images** from the Atari game environment as inputs. The pre-processed pixel input is **a summary of the game state  $s$** .
- The network outputs a vector containing **Q-values for each valid action**. The single output unit represents the  **$\hat{Q}$  function for a single action  $a$** .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Generalization: Deep Q-network (DQN)

The raw Atari 2600 frames are of size  $(210 \times 160 \times 3)$ . The pre-processing step adopted reduces the input dimensionality and deals with some artifacts of the game emulator.

- **Single frame encoding**: To encode a single frame, the maximum value for each pixel color value over the frame being encoded and the previous frame is returned.
- **Dimensionality reduction**: Extract the Y channel, also known as luminance, from the encoded RGB frame and rescale it to  $(84 \times 84 \times 1)$ .

The above pre-processing is applied to the 4 most recent raw RGB frames. Stacking together the recent frames as the game state can **transform the environment into a world closer to Markovian**.



# Generalization: Deep Q-network (DQN)

**Training algorithm for DQN:** The Q-network is learned by minimizing the mean squared error

$$J(\mathbf{w}) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})} [(y_t^{DQN} - \hat{Q}(s_t, a_t, \mathbf{w}))^2],$$

where  $y_t^{DQN}$  is the one-step ahead learning target

$$y_t^{DQN} = r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a', \mathbf{w}^-),$$

where  $\mathbf{w}^-$  represents the parameters of the target network, and the parameters  $\mathbf{w}$  of the online network are updated by sampling gradients from minibatches of past transition tuples  $(s_t, a_t, r_t, s_{t+1})$ .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen



# Generalization: Deep Q-network (DQN)

---

**Algorithm 2:** Deep Q-learning

---

Initialize replay memory  $D$  with a fixed capacity

Initialize action value function  $\hat{Q}$  with random weights  $\mathbf{w}$

Initialize target action value function  $\hat{Q}^-$  with weights  $\mathbf{w}^- = \mathbf{w}$

**for** episode  $k = 1, \dots, K$  **do**

    Observe initial frame  $x_1$  and pre-process frame to get state  $s_1$

**for** time step  $t = 1, \dots, T$  **do**

        Select action  $a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg \max_a \hat{Q}(s_t, a, \mathbf{w}) & \text{otherwise} \end{cases}$

        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$

        pre-process  $s_t, x_{t+1}$  to get  $s_{t+1}$ , and store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $D$

        Sample uniformly a random minibatch of  $N$  transitions

$\{(s_j, a_j, r_j, s_{j+1})\}_{j \in [N]}$  from  $D$

        Set  $y_j = r_j$  if episode ends at step  $j + 1$ , otherwise set

$y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a', \mathbf{w}^-)$

        Perform a stochastic gradient descent step on

$J(\mathbf{w}) = \frac{1}{N} \sum_{j=1}^N (y_j - \hat{Q}(s_j, a_j, \mathbf{w}))^2$  with respect to  $\mathbf{w}$

        Every  $C$  steps reset  $\mathbf{w}^- = \mathbf{w}$

# Experience Replay

To use large nonlinear function approximators and scale online Q-learning, DQN uses:

## 1. Experience Replay.:

- The agent's **experiences (the transitions)** at each time step  $e_t = (s_t, a_t, r_t, s_{t+1})$  is stored in a fixed-sized dataset  $D_t = \{e_1, \dots, e_t\}$ , known as the replay buffer. The replay buffer is used to store the most recent  $k = 1$  million experiences.
- The Q-network is updated by **SGD with sampled gradients from minibatch data**. Each transition sample in the minibatch is sampled uniformly at random from the pool of stored experiences,  $(s, a, r, s') \sim \text{Uniform}(D)$ .

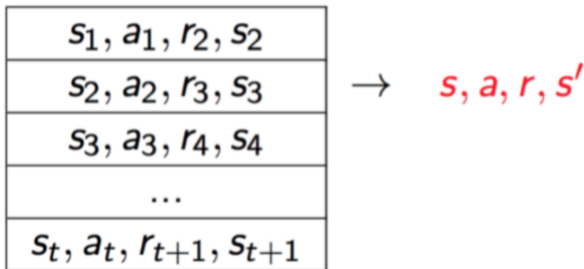


香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Experience Replay

Illustration of replay buffer. The transition  $(s, a, r, s')$  is uniformly sampled from the replay buffer for updating Q-network.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Experience Replay

Advantages over standard online Q-learning.

- **Greater data efficiency:** Each step of experience can be potentially used for many updates, which improves data efficiency.
- **Remove sample correlations:** Randomizing the transition experiences reduces the correlations between consecutive samples and therefore reduces the variance of updates and stabilizes the learning.
- **Avoiding oscillations or divergence:** The behavior distribution is averaged over many of its previous states and transitions, smoothing out learning and avoiding oscillations or divergence in the parameters.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Experience Replay

**Limitation of experience replay:** The replay buffer **does not differentiate important transitions or informative transitions** and it always overwrites with the recent transitions due to fixed buffer size. Similarly, the uniform sampling from the buffer gives equal importance to all stored experiences. A more sophisticated replay strategy, Prioritized Replay, **replays important transitions more frequently**, and therefore the agent learns more efficiently.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Target network

To use large nonlinear function approximators and scale online Q-learning, DQN uses:

## 2. Target network.:

- To further improve the stability of learning and deal with the non-stationary learning targets, a separate target network is used for generating the targets  $y_j$  in the Q-learning update.
- More specifically, for every  $C$  update steps the target network  $\hat{Q}(s, a, w^-)$  is updated by copying the parameters' values ( $w^- = w$ ) from the online network  $\hat{Q}(s, a, w)$ , and the target network remains unchanged and generates targets  $y_j$  for the following  $C$  updates.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Training details of DQN

- In the original DQN paper, a **different network** (or agent) was trained on **each game** with the same architecture, learning algorithm and hyperparameters.
- The authors **clipped** all **positive rewards from the game environment at +1** and all **negative rewards at -1**, which makes it possible to use the same learning rate across all different games.
- For games where there is a **life counter** (e.g. Breakout), the emulator also returns the number of lives left in the game, which is then used to mark the end of an episode during training by explicitly **setting future rewards to zeros**.



# Training details of DQN

- They also used a simple **frame-skipping technique** (known as action repeat): the agent selects actions on every 4-th frame instead of every frame, and its last action is repeated on skipped frames. This reduces the frequency of decisions without impacting the performance too much and enables the agent to **play roughly 4 times more games during training** (**less sample complexity, less reward delay**).





# Performance of Deep Q-network (DDQN)

## LETTER

doi:10.1038/nature14236

## Human-level control through deep reinforcement learning

Volodymyr Mnih<sup>1\*</sup>, Koray Kavukcuoglu<sup>1\*</sup>, David Silver<sup>1\*</sup>, Andrei A. Rusu<sup>1</sup>, Joel Veness<sup>1</sup>, Marc G. Bellemare<sup>1</sup>, Alex Graves<sup>1</sup>, Martin Riedmiller<sup>1</sup>, Andreas K. Fidjeland<sup>1</sup>, Georg Ostrovski<sup>1</sup>, Stig Petersen<sup>1</sup>, Charles Beattie<sup>1</sup>, Amir Sadik<sup>1</sup>, Ioannis Antonoglou<sup>1</sup>, Helen King<sup>1</sup>, Dharshan Kumaran<sup>1</sup>, Daan Wierstra<sup>1</sup>, Shane Legg<sup>1</sup> & Demis Hassabis<sup>1</sup>



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Performance of Deep Q-network (DDQN)

Atari 2600 games: Space Invaders



Atari 2600 games: Seaquest

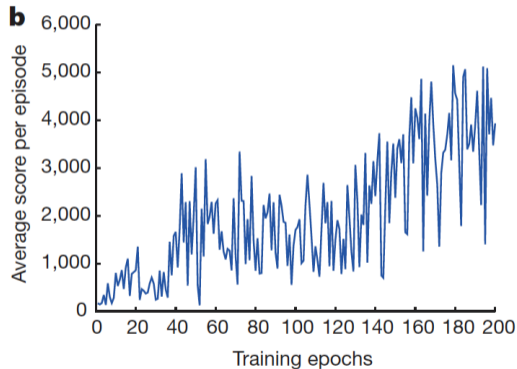
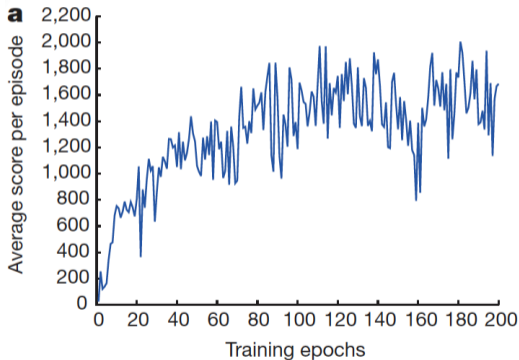


香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Performance of Deep Q-network (DDQN)

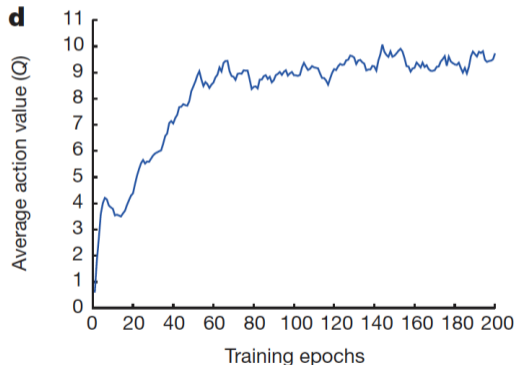
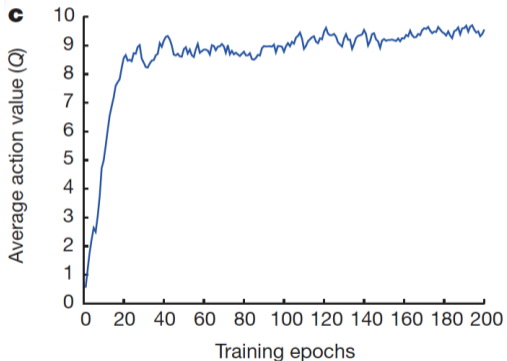
Average score for Space Invaders (Left)/ Seaquest (Right).



香港中文大学(深圳)  
The Chinese University of Hong Kong, Shenzhen

# Performance of Deep Q-network (DDQN)

Average predicted action-value for Space Invaders (Left)/ Seaquest (Right).



# Performance of Deep Q-network (DDQN)

Video Clips about Real Performance:

DQN in Space Invaders: [Link](#)



DQN in Seaquest: [Link](#)



香港中文大學(深圳)  
The Chinese University of Hong Kong, Shenzhen

# Performance of Deep Q-network (DDQN)

In summer, Deep Q-network agent can:

- receive only the pixels and the game score as inputs.
- surpass the performance of all previous algorithms.
- achieve a level comparable to that of a professional human games tester across a set of 49 games.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Reducing bias: Double deep Q-network (DDQN)

## Limitation of DQN:

- The max operator in DQN uses the same network values both to select and to evaluate an action.
- This setting makes it more likely to select overestimated values, resulting in overoptimistic target value estimates.

To prevent overestimation and reduce bias, we can decouple the action selection from action evaluation.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

## Reducing bias: Double deep Q-network (DDQN)

In double Q-learning, two action value functions are maintained. For computing targets, one function is used to **select the greedy action** and the other to **evaluate its value**

$$y_t^{DDQN} = r_t + \gamma \hat{Q}(s_{t+1}, \arg \max_{a'} \hat{Q}(s_{t+1}, a', w), w').$$

Alternatively, in double DQN, the greedy action is generated according to the online network with parameters  $w$ , but its value is estimated by the target network with parameters  $w^-$ . The computing of the target in DQN can be updated as:

$$y_t^{DDQN} = r_t + \gamma \hat{Q}(s_{t+1}, \arg \max_{a'} \hat{Q}(s_{t+1}, a', w), w^-).$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

The update to the target network stays unchanged from DQN.



# Reducing bias: Double deep Q-network (DDQN)

Published on AAAI 2016.

## Deep Reinforcement Learning with Double Q-learning

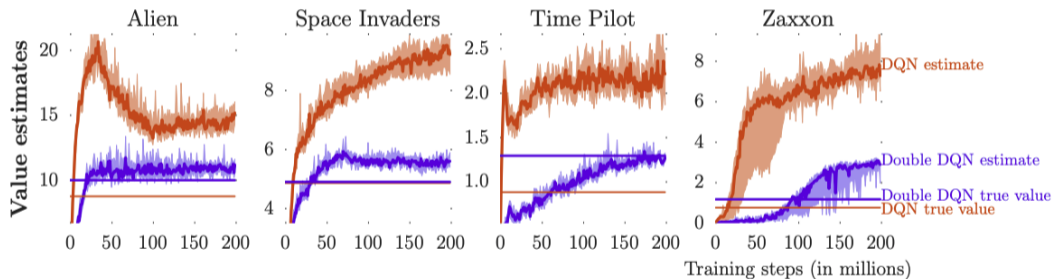
**Hado van Hasselt** and **Arthur Guez** and **David Silver**  
Google DeepMind



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

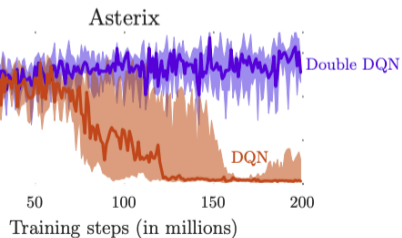
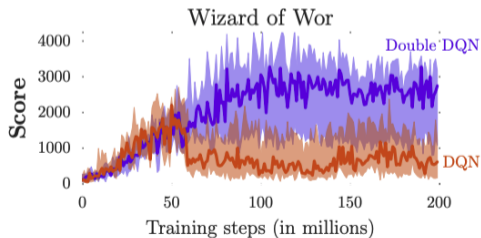
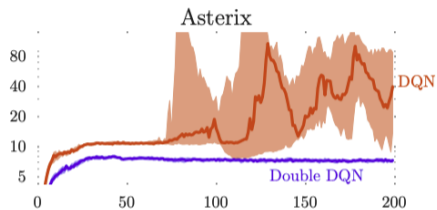
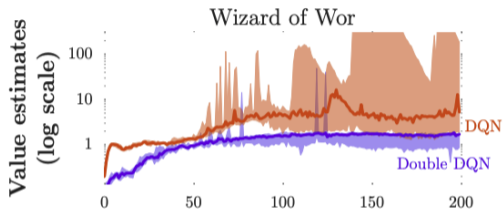
# Reducing bias: Double deep Q-network (DDQN)



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Reducing bias: Double deep Q-network (DDQN)



# Decoupling value and advantage: Dueling network

**Dueling network architecture.** Recall the advantage function, which relates the state value and the action value functions (assume following the same policy  $\pi$ )

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

Recall  $V^\pi(s) = \mathbb{E}_{a \sim \pi(s)}[Q^\pi(s, a)]$ , thus we have  $\mathbb{E}_{a \sim \pi(s)}[A^\pi(s, a)] = 0$ . Intuitively, the advantage function subtracts the value of the state from the Q-function to **get a relative measure of the importance of each action.**

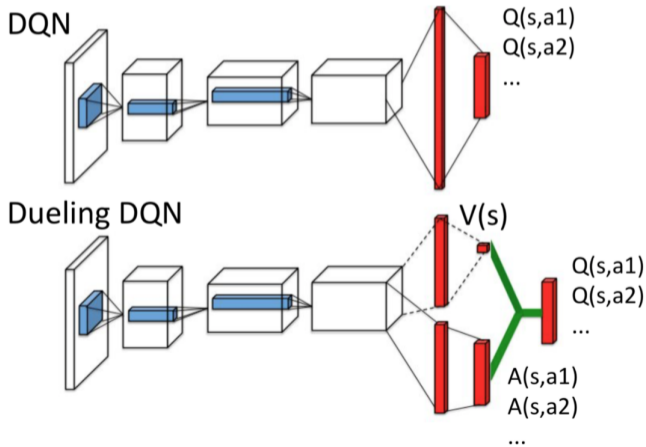


香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Decoupling value and advantage: Dueling network

**Dueling network architecture.** The dueling network has two streams to separately estimate the state value  $V(s)$  and the advantage  $A(s, a)$  for each action.



# Decoupling value and advantage: Dueling network

The separated two-stream design is based on the following observations or intuitions from the authors.

- For many states, it is unnecessary to estimate the value of each possible action choice. In some states, the action selection can be of great importance, but in many other states, the choice of action has no repercussions on what happens next. On the other hand, the state value estimation is of significant importance for every state for a bootstrapping-based algorithm like Q-learning.
- Features required to determine the value function may be different than those used to estimate action benefits accurately.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

## Q-value estimation in the dueling DQN

**Q-value estimation.** From the definition of the advantage function, we have

$$Q^\pi(s, a) = A^\pi(s, a) + V^\pi(s), \text{ and } \mathbb{E}_{a \sim \pi(s)}[A^\pi(s, a)] = 0$$

Furthermore, for a deterministic policy (commonly used in value-based deep RL)

$$a^* = \arg \max_{a' \in A} Q(s, a')$$

it follows that  $Q(s, a^*) = V(s)$  and hence  $A(s, a^*) = 0$ . The greedily selected action has zero advantage in this case.



## Q-value estimation in the dueling DQN

Now consider the dueling network architecture for function approximation:

- $\hat{V}(s, w, w_V)$ : the scalar output value function from one stream of the fully connected layers.
- $A(s, a, w, w_A)$ : the vector output advantage function from the other stream.
- $w$ : the shared parameters in the convolutional layers.
- $w_V$  and  $w_A$ : the parameters in the two different streams of fully connected layers.

The aggregating module is  $\hat{Q}(s, a, w, w_A, w_V) = \hat{V}(s, w, w_V) + A(s, a, w, w_A)$ .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen



## Q-value estimation in the dueling DQN

Given  $\hat{Q}$ , we cannot recover  $\hat{V}$  and  $A$  uniquely. To make the Q-function **identifiable**, we can force the advantage function to have zero estimate at the chosen action:

$$\hat{Q}(s, a, w, w_A, w_V) = \hat{V}(s, w, w_V) + \left( A(s, a, w, w_A) - \max_{a' \in A} A(s, a', w, w_A) \right). \quad (1)$$

Since  $a^* = \arg \max_{a' \in A} \hat{Q}(s, a', w, w_A, w_V) = \arg \max_{a' \in A} A(s, a', w, w_A)$ , and thus  $\hat{Q}(s, a^*, w, w_A, w_V) = \hat{V}(s, w, w_V)$ . Thus, the stream  $\hat{V}$  provides an estimate of the value function, and the other stream  $A$  generates advantage estimates.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

## Q-value estimation in the dueling DQN

To make the Q-function **identifiable**, an alternative aggregating module replaces the max operator with a mean operator

$$\hat{Q}(s, a, w, w_A, w_V) = \hat{V}(s, w, w_V) + \left( A(s, a, w, w_A) - \frac{1}{|A|} \sum_{a'} A(s, a', w, w_A) \right). \quad (1)$$

Although this design in some sense loses the original semantics of  $\hat{V}$  and  $A$ , the author argued that it improves the stability of learning.



# Q-value estimation in the dueling DQN

Published on International conference on machine learning (ICML) 2016.

---

## Dueling Network Architectures for Deep Reinforcement Learning

---

**Ziyu Wang**

**Tom Schaul**

**Matteo Hessel**

**Hado van Hasselt**

**Marc Lanctot**

**Nando de Freitas**

Google DeepMind, London, UK

ZIYU@GOOGLE.COM

SCHAUL@GOOGLE.COM

MTTHSS@GOOGLE.COM

HADO@GOOGLE.COM

LANCTOT@GOOGLE.COM

NANDODEFREITAS@GMAIL.COM



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

## Distributional Q-learning

An alternative approach to mitigate the overestimation effect is to write both the Q-function and the target into distributions in the Bellman optimality equation. Recall that the Q-value is defined as the expectation of the stochastic return (denote as  $Z^\pi$  to be distinguished from  $G$ )

$$Q^\pi(s, a) \stackrel{\text{def}}{=} \mathbb{E}_\pi[Z^\pi(s, a)] = \mathbb{E}_\pi[G_t | s_t = s, a_t = a] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right].$$

In the distributional setting, we can take away the expectation and consider the full distribution of the random variable  $Z^\pi$ . The distributional Bellman operator  $\mathcal{T}^\pi$  for  $Z$  is

$$\mathcal{T}^\pi Z(s, a) \stackrel{D}{=} R(s, a) + \gamma Z(s', a'),$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Distributional Q-learning

We are interested in the TD error between random variable  $Z^\pi$  and  $\mathcal{T}^\pi Z^\pi$ . To measure the distance between random variables, we adopt the  $p$ -Wasserstein metric  $d_p$ .

## Definition

Given two random variables  $U, V$  with their respective cumulative density functions  $F_U, F_V$ , the Wasserstein metric is defined as

$$d_p(U, V) = \left( \int_0^1 |F_U^{-1}(u) - F_V^{-1}(u)|^p du \right)^{1/p}.$$



# Distributional Q-learning

It is shown that  $\mathcal{T}^\pi$  is a contraction operator under the Wasserstein metric.

## Lemma

$\mathcal{T}^\pi : \mathcal{Z} \rightarrow \mathcal{Z}$  is a  $\gamma$ -contraction in  $d_p$ .

Note that however the contraction property of  $\mathcal{T}^\pi$  does not hold under KL divergence or total variation.

In general, optimality operators who have a fixed point  $Z^* = \mathcal{T}Z^*$  does not guarantee the convergence of the iteration  $Z_{k+1} \leftarrow \mathcal{T}Z_k$ . When the optimal policy is unique, this contraction operator guarantees a convergence. Despite that this assumption might not hold in general in practice, the practical performance of the algorithm has been satisfying.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Distributional Q-learning

## Lemma

*If the optimal policy is unique, then the iteration  $Z_{k+1} \leftarrow \mathcal{T}Z_k$  converges to  $Z^{\pi^*}$ .*

In its implementations,  $Z$  will be represented by a histogram. The update of the distributional Q-learning on the histogram is then

$$Z(s_t, a_t) \leftarrow (1 - \alpha_t)Z(s_t, a_t) + \alpha_t \Pi_C(R_t + \gamma Z(s_{t+1}, \pi_z(s_{t+1}))),$$

where  $\Pi_C$  is the projection operator to assign the probability density into the histogram bins and  $\alpha_t$  is the step size.



# Distributional Q-learning

Published on AAI 2018.

## Distributional Reinforcement Learning with Quantile Regression

**Will Dabney**  
DeepMind

**Mark Rowland**  
University of Cambridge\*

**Marc G. Bellemare**  
Google Brain

**Rémi Munos**  
DeepMind

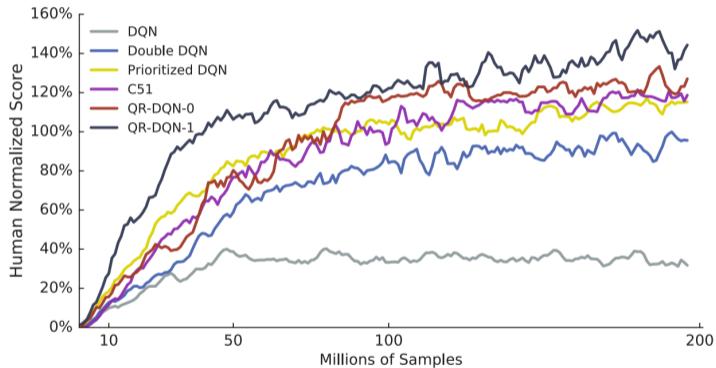


香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen



# Distributional Q-learning



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Question and Answering (Q&A)



香港中文大學(深圳)  
The Chinese University of Hong Kong, Shenzhen