UVA | SCHOOL of DATA SCIENCE

Extensions to Q-Learning

Reinforcement Learning
School of Data Science
University of Virginia

Last updated: February 27, 2024

Double Q-Learning

Main Idea

DQN requires target estimates of this form:

$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t)$$

The max() operation is used to estimate value
There may be noise in the system
Tends to produce a bias: overestimating value of Q

Paper: Deep Reinforcement Learning with Double Q-learning
Hado van Hasselt, Arthur Guez, David Silver. Google DeepMind

Main Idea, contd.

Double DQN

The idea of Double Q-learning is to reduce overestimations by decomposing the max operation in the target into action selection and action evaluation. Although not fully decoupled, the target network in the DQN architecture provides a natural candidate for the second value function, without having to introduce additional networks. We therefore propose to evaluate the greedy policy according to the online network, but using the target network to estimate its value. In reference to both Double Q-learning and DQN, we refer to the resulting algorithm as Double DQN. Its update is the same as for DQN, but replacing the target Y_t^{DQN} with

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t)$$

Decompose max() operation into action selection, action evaluation

Notice there are two Qs

- One determines greedy policy using online network $\boldsymbol{\theta}_t$
- Another fairly evaluates the policy using the target network $\boldsymbol{\theta}'_t$

Double Q-Learning Algorithm

Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau << 1$
for each iteration **do**
  for each environment step **do**
    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$
    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
  for each update step **do**
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute target Q value:
      $Q^*(s_t, a_t) \approx r_t + \gamma\ Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$
    Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$
    Update target network parameters:
      $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$

Bias Estimates

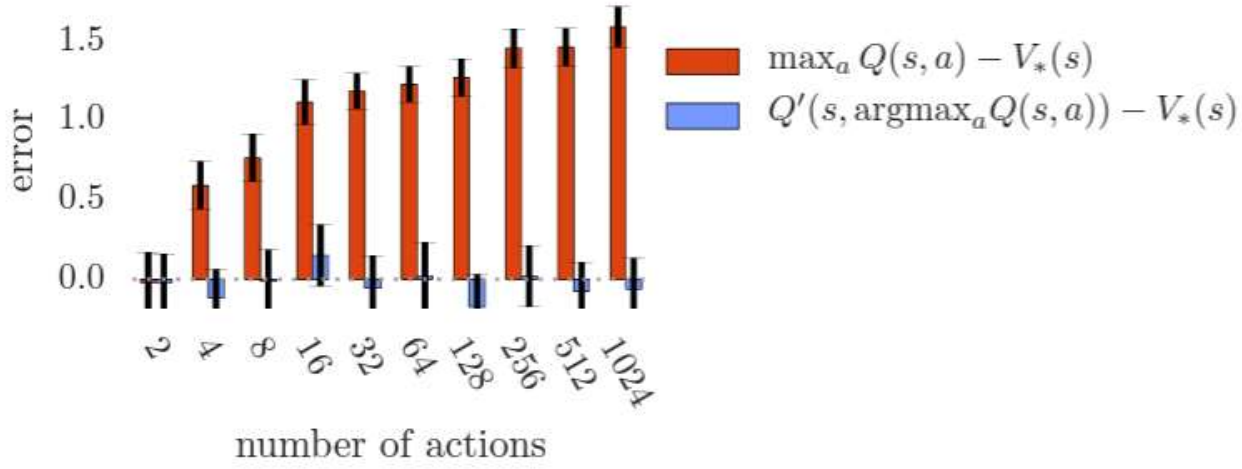
Theorem 1. *Consider a state s in which all the true optimal action values are equal at $Q_*(s,a) = V_*(s)$ for some $V_*(s)$. Let Q_t be arbitrary value estimates that are on the whole unbiased in the sense that $\sum_a (Q_t(s,a) - V_*(s)) = 0$, but that are not all correct, such that $\frac{1}{m}\sum_a (Q_t(s,a) - V_*(s))^2 = C$ for some $C > 0$, where $m \geq 2$ is the number of actions in s. Under these conditions, $\max_a Q_t(s,a) \geq V_*(s) + \sqrt{\frac{C}{m-1}}$. This lower bound is tight. Under the same conditions, the lower bound on the absolute error of the Double Q-learning estimate is zero. (Proof in appendix.)*

=> The max() operator induces upward bias that shrinks with increasing #actions, but does not vanish

Bias Estimates, contd.

Overoptimism increases with number of actions (red bars)

However, for Double Q-Learning, bias remains small

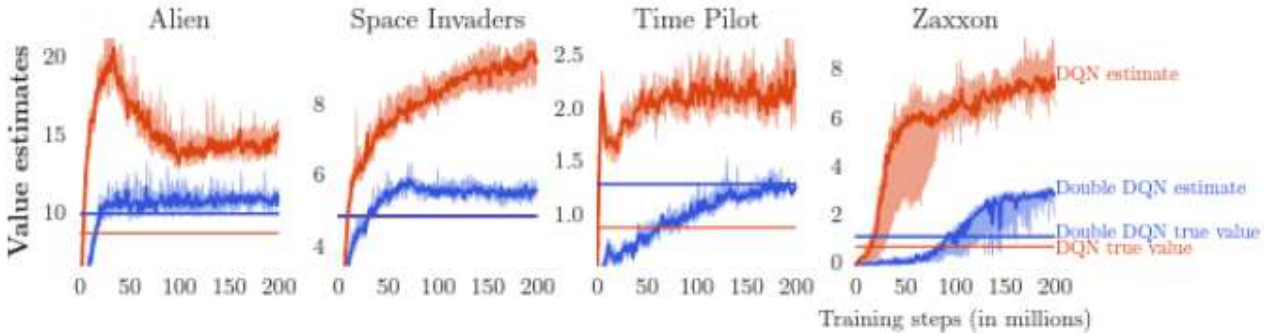


Bias Estimates - Atari

Horizontal lines are actual discounted value of best learned policy (unbiased)

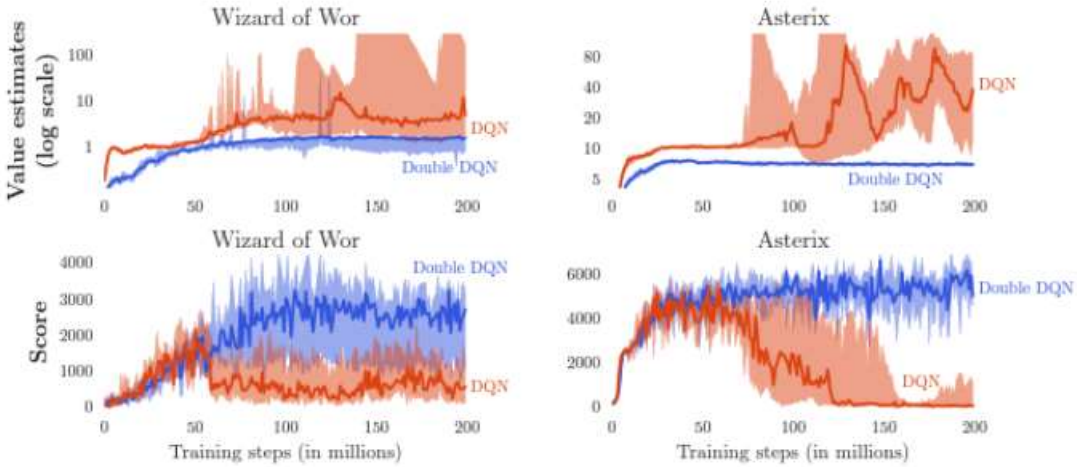
Curves are estimates, which exhibit bias

Double DQN best policy (blue horiz. line) outperforms DQN best policy (red horiz. line)



Bias Estimates – Atari, Extreme Cases

In these cases, overoptimism in DQN is extreme (red curve)
Where red separates from blue, bias increases
This has detrimental effect on score (Double Q-Learning outperforms)
Double Q-Learning is also more stable



Findings

```
target = (reward + self.gamma *
          np.amax(self.model.predict(next_state)[0]))
```

The bias resulting from max() in DQN can be large

Double DQN can be more stable and reliable than DQN

Double DQN can find better policies than DQN

```
t = self.target_model.predict(next_state)[0]
target[0][action] = reward + self.gamma * np.amax(t)
```

Dueling Deep Q-Network

Dueling Networks

Deep RL before this paper used conventional architectures (CNN, LSTM)

Focus here is new architecture better suited to model-free RL

Dueling architecture separates state values and action advantages

Paper: Dueling Network Architectures for Deep Reinforcement Learning
Wang et. al.

Value and Advantage Function Definitions

Value functions:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}\left[R_t \middle| s_t = s, a_t = a, \pi\right] \\ V^{\pi}(s) &= \mathbb{E}_{a\sim\pi(s)}\left[Q^{\pi}(s,a)\right]. \end{aligned}$$

The advantage function isolates effect of action taken

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

V(*s*) measures how good it is to be in state *s*
A(s,a) measures relative importance of each action

Sometimes, the action doesn't matter

Dueling Architecture

Separate streams for value and advantage functions

Common convolution feature

Top figure is Q-network

Bottom figure is dueling Q-network

Value and advantage streams are combined

For each network, outputs are Q(s,a)

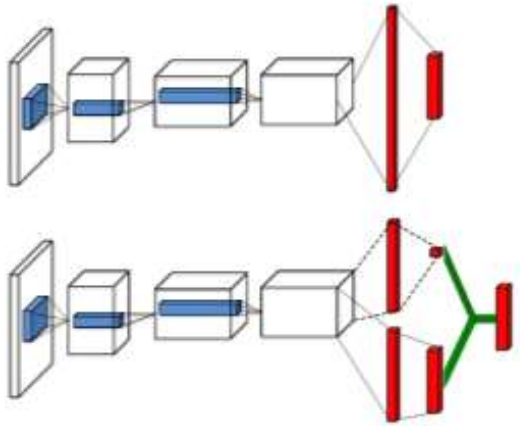


Figure 1. A popular single stream Q-network (**top**) and the dueling Q-network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q-values for each action.

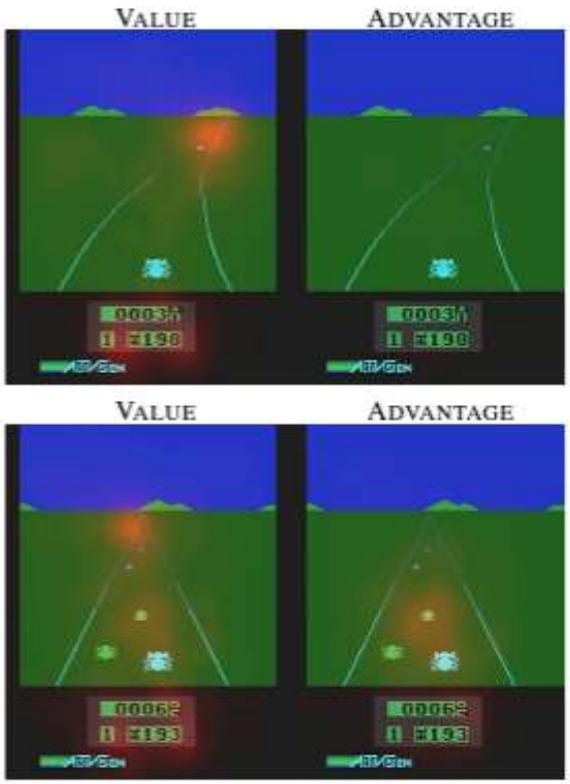
Value and Advantage Function Saliency

Figure shows saliency map for two time steps

Value stream pays attention to horizon and score

Advantage stream: when no cars are on road, action doesn't matter (no attention paid)

When cars are on road, advantage stream pays attention to car in front



Identifiability

Given Q = V + A, we cannot recover V and A uniquely
(adding any c to V, and subtracting c from A, leaves Q unchanged)

Add a constraint: subtract the average advantage from A:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'; \theta, \alpha) \right)$$

For actions with above average advantage, the second term in (*) will be positive
For action with greatest advantage, (*) term will be largest across all actions

Review of Dueling Architecture Code

We can see an implementation of Dueling Q-Network here:

Paper: *Reinforcement Learning for optimal sepsis treatment policies* (2017)
Authors: Raghu, Komorowski, Ahmed, Celi, Szolovits, Ghassemi

GitHub repo:
https://github.com/aniruddhraghu/sepsisrl/blob/master/continuous/q_network.ipynb

See section:
advantage and value streams

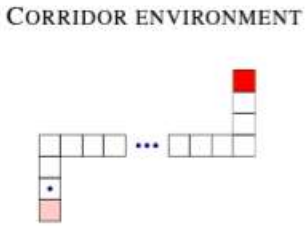
Implementation

Our two streams (V, A) for Q are part of the model architecture

Training step runs the same as a standard Q-network: backpropagation

Evaluation

Sets up simple environment called *corridor*.
Redness of state signifies reward



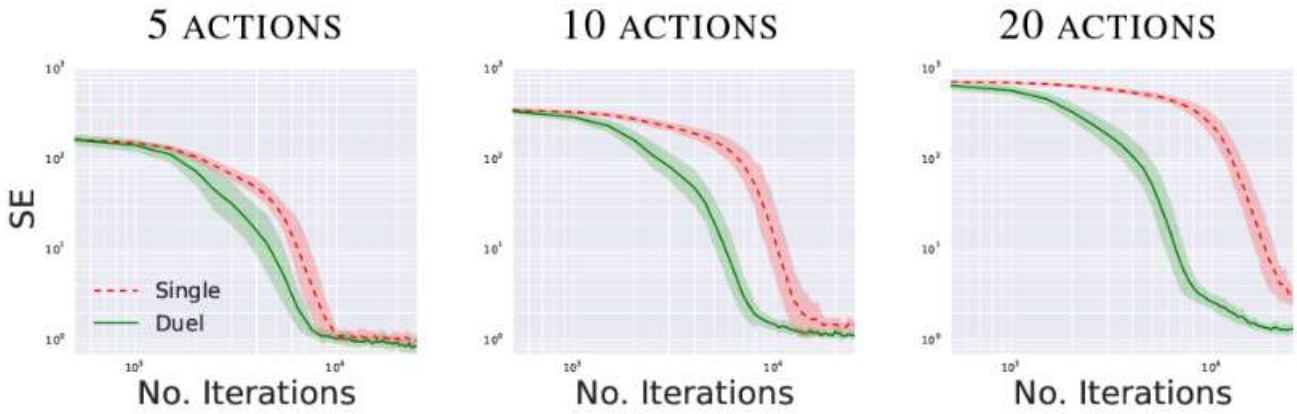
Can move up, down, left, right, or no move.

ε-greedy policy, measure performance by squared error (SE) vs. true action values

Single and Duel networks use MLP with three layers

For duel network, after first hidden layer, network branches off to two streams

Evaluation, contd.



Dueling network converges faster than Single

More pronounced as |A| increases

As |A| increases, SE is lower for Dueling network. It performs better.