

Reinforcement Learning

Exercise 5: Value-function Approximation

26.05.2023

Sebastian Rietsch

Exercise Sheet 4

TD Control

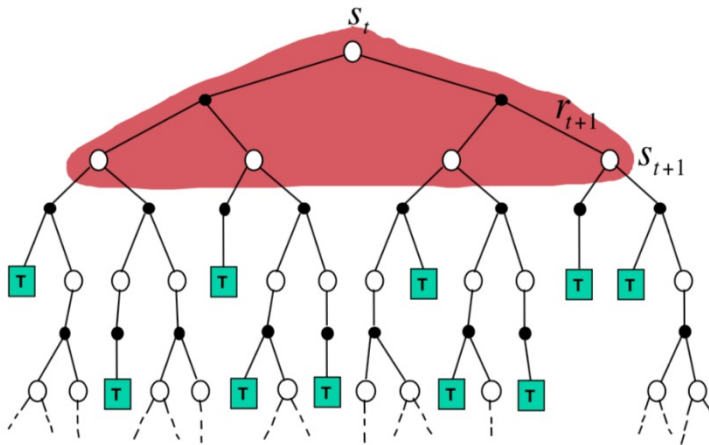


Exercise Sheet 3

Temporal Difference Learning

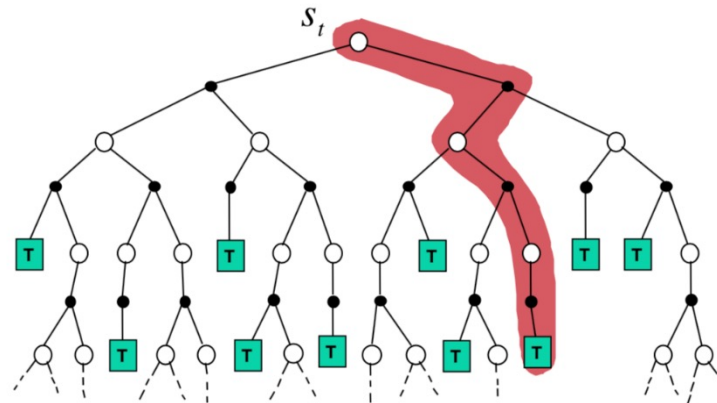
DP Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi [R_{t+1} + \gamma V(S_{t+1})]$$



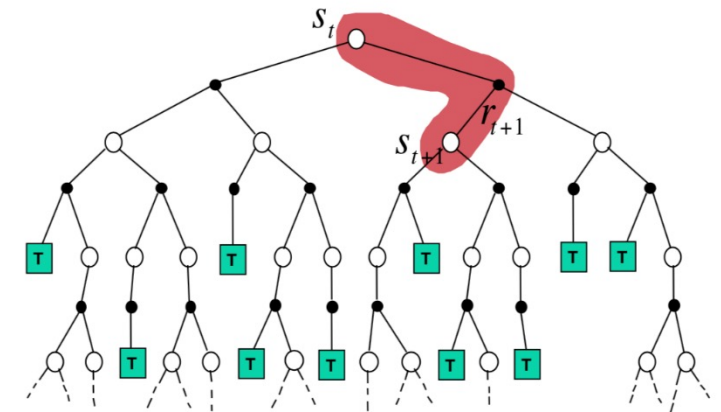
MC Backup

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$



TD Backup

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

SARSA

on-policy control

- Apply TD to $Q(s, a)$
- Use ϵ -greedy policy improvement
- Update at every time-step

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$

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

Loop for each episode:

 Initialize S

 Choose A from S using policy derived from Q (e.g., ϵ -greedy)

 Loop for each step of episode:

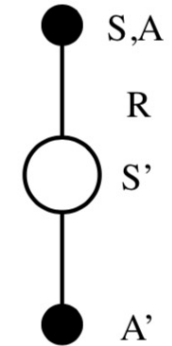
 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ϵ -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

 until S is terminal



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Q-Learning

off-policy control

- Evaluate one policy while following another
- Can re-use experience gathered from old policies

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

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

Loop for each episode:

 Initialize S

 Loop for each step of episode:

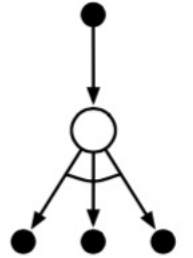
 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

 until S is terminal



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Exercise 5

PyTorch Intro



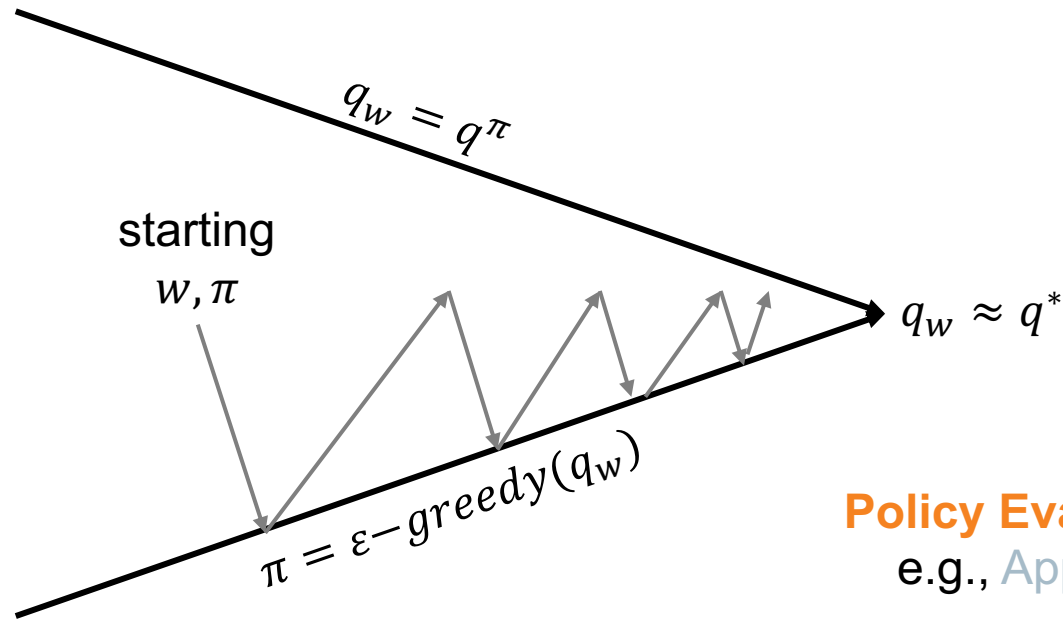
Exercise Sheet 6

Value Function Approximation



Value Function Approximation

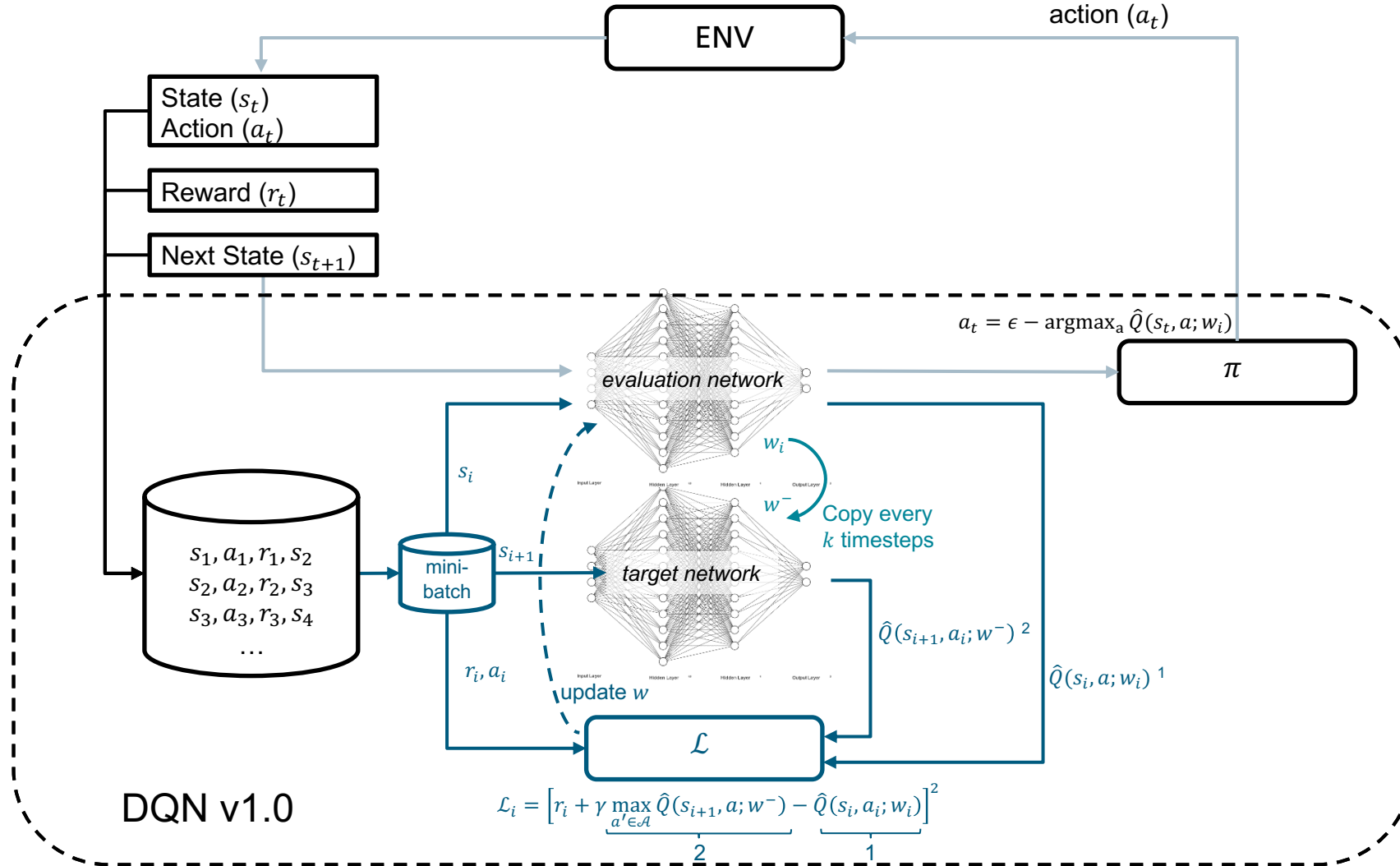
- Our goal is to learn good parameters w that approximate the true value function well:



Policy Evaluation: Estimate $\hat{q}(\cdot, \cdot, w) \approx q_\pi$
e.g., Approximate Policy Evaluation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., ϵ -greedy Policy Improvement

Deep Q-Networks (DQNs)



(Double) DQNs

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

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

For episode = 1, M **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1, T$ **do**

With probability ε select a random action a_t

otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$

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

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ

Every C steps reset $\hat{Q} = Q$

End For

End For

Algorithm 1 Double Q-learning

```
1: Initialize  $Q^A, Q^B, s$ 
2: repeat
3:   Choose  $a$ , based on  $Q^A(s, \cdot)$  and  $Q^B(s, \cdot)$ , observe  $r, s'$ 
4:   Choose (e.g. random) either UPDATE(A) or UPDATE(B)
5:   if UPDATE(A) then
6:     Define  $a^* = \operatorname{argmax}_a Q^A(s', a)$ 
7:      $Q^A(s, a) \leftarrow Q^A(s, a) + \alpha(s, a) (r + \gamma Q^B(s', a^*) - Q^A(s, a))$ 
8:   else if UPDATE(B) then
9:     Define  $b^* = \operatorname{argmax}_a Q^B(s', a)$ 
10:     $Q^B(s, a) \leftarrow Q^B(s, a) + \alpha(s, a) (r + \gamma Q^A(s', b^*) - Q^B(s, a))$ 
11:  end if
12:   $s \leftarrow s'$ 
13: until end
```

Double DQNs

$$Y_t^{\text{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-).$$

$$Y_t^{\text{Q}} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}} Q(S_{t+1}, a; \theta_t); \theta_t).$$

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}} Q(S_{t+1}, a; \theta_t); \theta_t').$$

Double DQNs

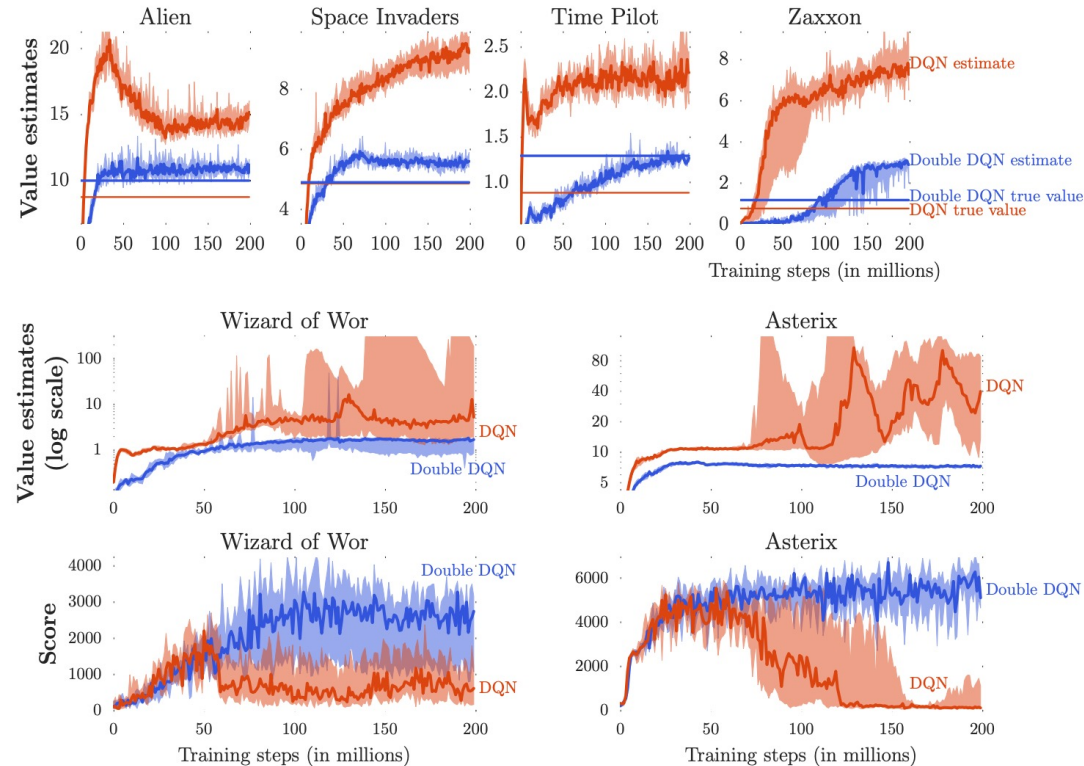


Figure 3: The **top** and **middle** rows show value estimates by DQN (orange) and Double DQN (blue) on six Atari games. The results are obtained by running DQN and Double DQN with 6 different random seeds with the hyper-parameters employed by Mnih et al. (2015). The darker line shows the median over seeds and we average the two extreme values to obtain the shaded area (i.e., 10% and 90% quantiles with linear interpolation). The straight horizontal orange (for DQN) and blue (for Double DQN) lines in the top row are computed by running the corresponding agents after learning concluded, and averaging the actual discounted return obtained from each visited state. These straight lines would match the learning curves at the right side of the plots if there is no bias. The **middle** row shows the value estimates (in log scale) for two games in which DQN's overoptimism is quite extreme. The **bottom** row shows the detrimental effect of this on the score achieved by the agent as it is evaluated during training: the scores drop when the overestimations begin. Learning with Double DQN is much more stable.

Thank you for your attention!