DQN

Blitz Course

Odalric-Ambrym Maillard, Fabien Pesquerel and Patrick Saux

April 5, 2021

Reminder: Q-Learning

- Observe transition (s, a, r, s').
- $a^* \in \arg\max_{a \in \mathcal{A}} Q(s', a)$.
- Optimal Bellman equation:

$$Q(s,a) = \mathbb{E}\Big[r + \gamma Q(s', a^*)\Big].$$

- Q-Learning is the **sampled version**:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s', a^*) - Q(s,a)\Big).$$

↳ Repeat over multiple episodes, possibly with forced exploration.

Q-Learning with function approximation

- What if state or action space is very large or infinite?
- Need to **generalise** a given policy from a small sample of observed states and actions.
- Example: autonomous car
  - If you know what happens to the car when you move the wheel by $10°$ and $20°$, you can guess the effect of moving it by $15°$.

↳ Learn $Q_\theta(s, a)$ an approximation of Q with parameter $\theta \in \mathbb{R}^p$ by minimising the empirical Bellman error

$$\min_{\theta \in \mathbb{R}^p} \sum_{(s,a,r,s')} \left(Q_\theta(s, a) - r - \gamma \max_{a' \in \mathcal{A}} Q_\theta(s', a') \right)^2 .$$

Drawbacks

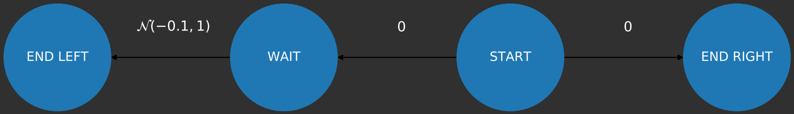
- ✓ Exact Q-Learning is provably convergent (under mild assumptions)...
- ✗ ... much less is known with function approximation (can diverge in some cases).
- ✗ ... can perform poorly in finite time:
  - Initially Q_θ is very noisy.
  - $\max_{a' \in \mathcal{A}} Q_\theta(s', a')$ is biased towards overestimated actions.
- ↳ Decouple **selection** and **evaluation**.
- ✗ Needs "good" (in theory i.i.d) sample (s, a, r, s') to train Q_θ... but (s, a, r, s') depends on previous Q_θ!
- ↳ Reuse past experiences.

Double Q-Learning

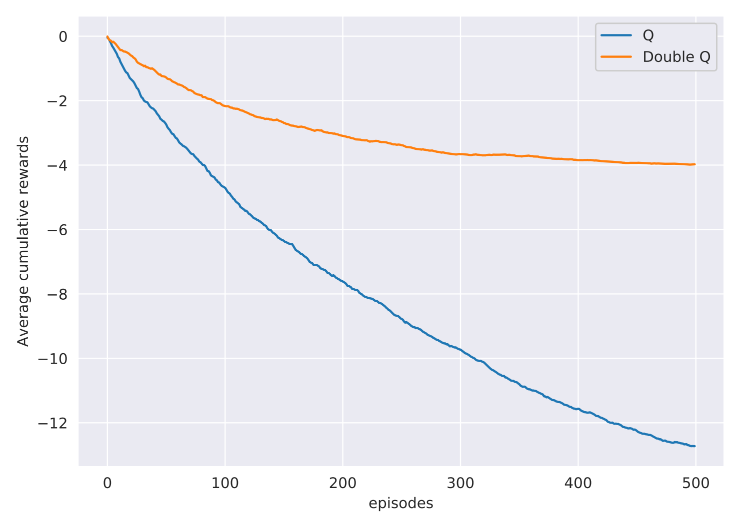
- Observe transition (s, a, r, s').
- Two Q-functions Q^A and Q^B.
- At each turn, choose randomly between either
  - $a^* \in \arg\max_{a \in \mathcal{A}} Q^A(s', a)$,
  - $Q^A(s, a) \leftarrow Q^A(s, a) + \alpha\Big(r + \gamma Q^B(s', a^*) - Q^A(s, a)\Big)$,
- or
  - $b^* \in \arg\max_{b \in \mathcal{A}} Q^B(s', b)$.
  - $Q^B(s, a) \leftarrow Q^B(s, a) + \alpha\Big(r + \gamma Q^A(s', b^*) - Q^B(s, a)\Big)$.

Toy MDP

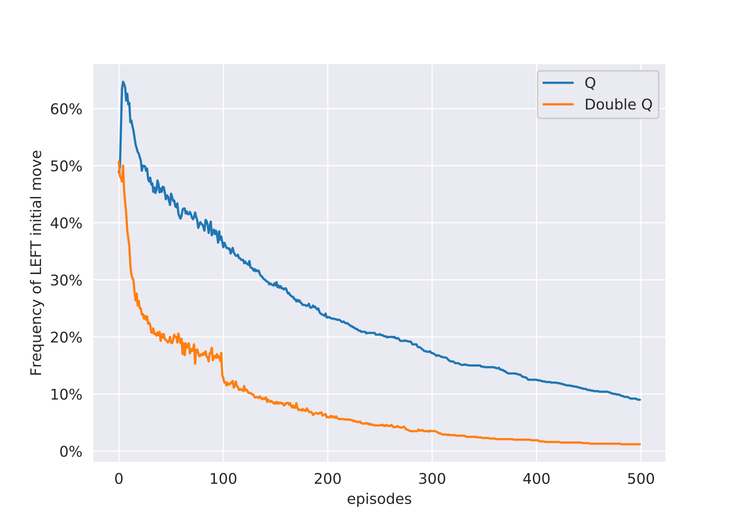
4 states MDP



Q-Learning vs Double Q-Learning



Q-Learning vs Double Q-Learning



Experience Replay

- Observe transition (s, a, r, s').
- Add it to **replay buffer** $\mathbb{B} \leftarrow \mathbb{B} \cup \{(s, a, r, s')\}$.
- Sample a random minibatch $\mathcal{B} \subset \mathbb{B}$.
- Minimise minibatch loss:

$$\min_{\theta \in \mathbb{R}^p} \sum_{(s,a,r,s') \in \mathcal{B}} \mathcal{L}_\theta(s, a, r, s').$$

↳ $\mathcal{B}$ "looks more" i.i.d.

↳ Extract more from each transition as they are used more than once: better sample efficiency.

DQN™

- **Replay buffer**: observe $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ and store it.
- **Greedy with exploration**: $a_t \in \arg\max_{a \in \mathcal{A}} Q_\theta(s_t, a)$ with probability $1 - \epsilon_t$, otherwise pick a random action.
- **Bellman loss**: sample minibatch $\mathcal{B}$ from replay buffer,

$$L_\theta = \sum_{(s,a,r,s') \in \mathcal{B}} \Big(\underbrace{Q_\theta(s,a)}_{\text{Q-network}} - r - \gamma \max_{a' \in \mathcal{A}} \underbrace{Q_{\theta^-}(s', a')}_{\text{target network}} \Big)^2 .$$

- Train θ (but not θ^-) with (a variant of) gradient descent

$$\theta \leftarrow \theta - \eta \nabla_\theta L_\theta .$$

- Every N transitions, update target parameters: $\theta^- \leftarrow \theta$.

DQN™

- ✓ Better convergence behaviour of Q_θ with experience replay.
- ✓ More stable policies with infrequent updates of the target network θ^- (although it is not really double Q-learning since θ^- is only frozen, not alternatively switched with θ; a variant called Double DQN does that).
- ↳ Patented by DeepMind around 2015 to reach human level performances on Atari 2600.