1 DQN

Deep Q Learning

1.1 Tabular (non-deep) case

Imagine that we have an environment that can be described as follows:

- It has an initial state (env.reset())
- It awaits actions from the user and depending on the action given by the user it samples next state from a probability distribution that depends only on the current state and the action taken by the user (Markovian)

Consider a function

$$Q_k(\text{state}, \text{action}) =$$
$$\mathbb{E}_{\text{next\_state} \sim P(\text{next\_state}|\text{state},\text{action})} \left[r_{\text{state},\text{next\_state}} + \gamma \max_a Q_{k-1}(\text{next\_state}, a) \right]$$

With

$$\forall_{s,a} Q_0(s,a) = 0$$

That score is discounted using $\gamma \leq 1$ (later rewards weight less, that is first reward is multiplied by γ^0 second one by γ^1 and so on)

Observe that for k we have that $Q_k(\text{state}, \text{action})$ gives us the best expected score that we can achieve when starting in state and making k moves with first one being action.
Proof by induction:
For $k = 0$
We have $\forall_{s,a} Q_0(s,a) = 0$
No moves no points.

For $k > 0$
We have that Q_{k-1} has this property therefore after making the first move the highest expected discounted score that we can obtain in exactly $k - 1$ moves is $\gamma \max_a Q_{k-1}(\text{next\_state}, a)$

Note that from the above we have in fact a method for calculating Q_k when we know the environment.

Now assume $\gamma < 1$.
We will show that Q_∞ exists (end of the game can be modelled by a special state, such that after we reach it we will only get rewards that are equal to 0).

We start with any:

$$Q, Q' : (S \times A) \to \mathbb{R}$$

and let us define

$$\tau : ((S \times A) \to \mathbb{R}) \to ((S \times A) \to \mathbb{R})$$

$$\tau(Q) = \lambda s, a. \mathbb{E}_{s' \sim P(s'|s,a)} \left[r_{s,s'} + \gamma \max_{a'} Q(s', a') \right]$$

Consider

$$\max_{s,a} |\tau(Q)(s, a) - \tau(Q')(s, a)|$$

$$\max_{s,a} |\tau(Q)(s, a) - \tau(Q')(s, a)| =$$
$$\max_{s,a} |\mathbb{E}_{s' \sim P(s'|s,a)} \left[r_{s,s'} + \gamma \max_{a'} Q(s', a') \right] -$$
$$\mathbb{E}_{s' \sim P(s'|s,a)} \left[r_{s,s'} + \gamma \max_{a'} Q'(s', a') \right] | =$$
$$\max_{s,a} \gamma |\mathbb{E}_{s' \sim P(s'|s,a)} \left[\max_{a'} Q(s', a') - \max_{a'} Q'(s', a') \right] |$$

Now for each s, a we can leave s' that gives the highest absolute difference.

$$\gamma |\mathbb{E}_{s' \sim P(s'|s,a)} \left[\max_{a'} Q(s', a') - \max_{a'} Q'(s', a') \right] | \leq$$
$$\gamma \max_{s'} \left| \max_{a'} Q(s', a') - \max_{a'} Q'(s', a') \right|$$

Now we have that $\max_{a'} Q(s', a') - \max_{a'} Q'(s', a')$ is either non-negative or negative - wlog assume the first and note that $\max_{a'} Q(s', a') - \max_{a'} Q'(s', a') \leq \max_{a'} Q(s', a') - Q'(s', a')$

Using this we can write

$$\max_{s,a} |\tau(Q)(s, a) - \tau(Q')(s, a)| \leq \gamma \max_{s,a} |Q(s, a) - Q'(s, a)|$$

Now rest follows from the Banach fixed point theorem Let

$$Q, Q' : (S \times A) \to \mathbb{R}$$

$$\|\tau^n(Q) - \tau^n(Q')\|_\infty = \max_{s,a} |\tau^n(Q)(s, a) - \tau^n(Q')(s, a)|$$

1.2 Deep case

Use two networks

$$Q, Q' : (S \times A) \to \mathbb{R}$$

Training algorithm:

- freeze Q' - that is do not propagate gradients to Q'
- act using strategy that utilizes Q (for example 0.95 of actions from Q other actions random, possibly less random actions as Q gets better)
- save tuples (state, action, reward, next_state) from environment to a buffer
- train Q as follows
  - Sample a batch of tuples from the buffer
  - For each tuple (s, a, r, s') we want $\|Q(s, a) - (r_{s,s'} + \max_{a'} \gamma Q'(s, a'))\|_2$ to be minimized
- each x steps make Q' to be a frozen copy of Q