Page 9, N-steps DQN

$$Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, a_{t+1})$$

This equation is recursive, which means that we can express $Q(s_{t+1}, a_{t+1})$ in terms of itself, which gives us this result:

$$Q(s_t, a_t) = r_t + \gamma \max_a [r_{a,t+1} + \gamma \max_{a'} Q(s_{t+2}, a')]$$

Value $r_{a,t+1}$ means local reward at time t+1, after issuing action a. However, if we assume that our action a at the step t+1 was chosen optimally, or close to optimally, we can omit $\max_a$ operation and obtain this:

$$Q(s_t, a_t) = r_t + \gamma r_{t+1} + \gamma^2 \max_{a'} Q(s_{t+2}, a')$$

Page 10

So, what happens in a one-step case? We have three total updates possible (we don't use max, as there is only one action available):

1. $Q(s_1, a) \leftarrow r_1 + \gamma Q(s_2, a)$
2. $Q(s_2, a) \leftarrow r_2 + \gamma Q(s_3, a)$
3. $Q(s_3, a) \leftarrow r_3$

Now let's consider a two-step case. This situation again has three updates:

1. $Q(s_1, a) \leftarrow r_1 + \gamma r_2 + \gamma^2 Q(s_3, a)$
2. $Q(s_2, a) \leftarrow r_2 + \gamma r_3$
3. $Q(s_3, a) \leftarrow r_3$

Page 13, Double DQN

In the basic DQN, our target value for Q looked like this:

$$Q(s_t, a_t) = r_t + \gamma \max_a Q'(s_{t+1}, a)$$

$Q'(s_{t+1}, a)$ was Q-values calculated using our target network, so we update with the trained network every n steps. The authors of the paper proposed choosing actions for the next state using the trained network but taking values of Q from the target net. So, the new expression for target Q-values will look like this:

$$Q(s_t, a_t) = r_t + \gamma \max_a Q'(s_{t+1}, \arg\max_a Q(s_{t+1}, a))$$

Page 21, Prioritized replay buffer

From the mathematical point of view, priority of every sample in the buffer is calculated as $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$, where p_i is the priority of i-th sample in the buffer and α is the number which shows

To get this, the authors used sample weights, which need to be multiplied to the individual sample loss. The value of weight for each sample is defined as $w_i = (N \cdot P(i))^{-\beta}$, where β is another hyperparameter, which should be between 0 and 1.

Page 27, Dueling DQN

This constraint could be enforced in various ways, for example, via the loss function, but in the above paper, the authors proposed a very elegant solution by subtracting from the Q expression in the network mean value of the advantage, which effectively pulls the mean for advantage to the zero: $Q(s,a) = V(s) + A(s,a) - \frac{1}{N}\sum_k A(s,k)$.

Page 31, Categorical DQN

As a next step, the authors have shown that Bellman equation can be generalized for distribution case and it will have a form $Z(x,a) \stackrel{D}{=} R(x,a) + \gamma Z(x',a')$, which is very similar to familiar Bellman equation, but now Z(x, a), R(x, a) are the probability distributions and not numbers. The effect of this equation on distribution is demonstrated on the plots below, taken from the original paper.

We have original distribution of values shown in upper-left corner. After multiplication on γ (upper-right chart)

Page 39

At the end of the function, we need to compute output of network and calculate KL-divergence between projected distribution and networks output for the taken actions. KL divergence shows how much two distributions differ and is defined as $D_{KL}(P\|Q) = -\sum_i p_i \log q_i$