Double Learning

Transition Probability, $P(s, a, s')$

Return, $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$

$V^*(s) = \max_a \sum_{s'} P(s'|s,a)\left(r + \gamma V^*(s')\right)$

$V^*_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[r + \gamma V^*_k(s')]$

$Q^*(s,a) = \sum_{s'} p(s'|s,a)\left[r + \gamma \max_{a'} Q^*(s',a')\right]$

$Q_{i+1}(s) \leftarrow \sum_{s'} P(s'|s,a,)\left[r + \gamma \cdot \max_{a'} Q_i(s',a')\right]$

Policy Evaluation

$V^{\pi_k}_{i+1}(s) = \sum_{s'} P(s'|s,\pi_k(s))\left[r + \gamma V^{\pi_k}_i(s')\right]$

Policy Improvement

$\pi_{k+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s,a)[r + \gamma V^{\pi_k}(s')]$

Q – Learning:

$Q_{i+1}(s,a) \leftarrow Q_i(s,a) + \alpha\left(r + \gamma \cdot \max_{a'} Q_i(s',a') - Q_i(s,a)\right)$

DQN Update:

$\theta_{t+1} = \theta_t + \alpha\left(y_t - Q(s_t,a_t;\theta)\right)\nabla_{\theta_t} Q(s_t,a_t;\theta_t)$

DQN Q Update:

$Q(s_t,a_t;\theta_{t+1}) \approx Q(s_t,a_t;\theta) + \alpha\left(y_t - Q(s_t,a_t;\theta_t)\right)\left|\left|\nabla_{\theta_t} Q(s_t,a_t;\theta_t)\right|\right|_2^2$

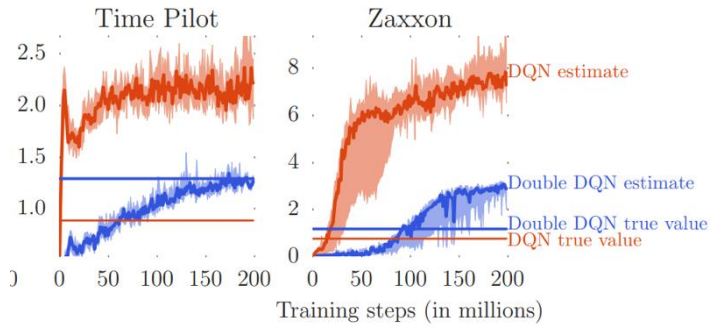
$$y_i = r_i + \gamma \cdot \max_{a'} Q_\theta(s'_i, a'_i)$$

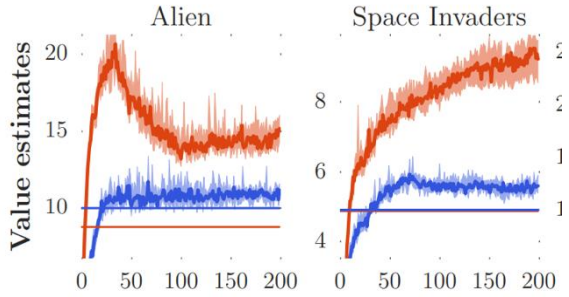
- Double Q – Learning
  - Decouple action selection from evaluation.
  - $y_t = r + \gamma Q\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a)\right)$
  - $y^D_t = r + \gamma Q'\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a)\right)$ → 2 value functions

Train 2 independent Q-networks: Q_θ & $Q_{\theta'}$

- Double Deep Q -Learning
  - $y^D_t = r + \gamma Q\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta'_t\right)$ → target network
- Q_θ : Trained to select actions
- $Q_{\theta'}$: Trained to evaluate action

- **NOTE:**
  When combined with the target network trick to avoid shifting targets: Four networks will have to be trained.





Algorithm 1: Double Deep Q – Learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $\mathcal{N}$
Initialize $Q(\theta)$ & $Q'(\theta')$ with random weights
For episode = 1, M **do**
  Initialize sequence s_1
  For t = 1, T **do**
    with probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$
    Execute action a_t in emulator and observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & ; for\ terminal\ s_{j+1} \\ r_j + \gamma Q\left(s_{j+1}, \arg\max_a Q(s_{j+1}, a; \theta); \theta'\right) & ; for\ non-terminal\ s_{j+1} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(s_j, a_j; \theta)\right)^2$
  end For
end For

https://www.pratikumar.co.in/

Van Hasselt, Hado, Arthur Guez, and David Silver, "Deep reinforcement learning with double q – learning." arXiv preprint arXiv: 1509.06461 (2015)