Reinforcement Learning Lab 3: Pseudocode for DQN

Min-hwan Oh

Algorithm 1 DQN

INPUT: target network update period τ, total number of episodes E, initial time steps before update init, learning rate α, exploration prob ϵ, batchsize for training N
INITIALIZE: DQN principal network $Q_\theta(s, a)$ with parameters θ, target network $Q_{\theta^-}(s, a)$ with parameters θ^-, time steps counter *counter* $\leftarrow 0$, empty buffer $R \leftarrow \{\}$
for $e = 1, 2, 3 ... E$ **do**
    while episode not terminated **do**
        Execute actions
        counter $\leftarrow$ *counter* $+ 1$
        Given state s_t, for prob ϵ, take action uniformly random; otherwise, take action by being greedy $a_t \leftarrow \arg\max_a Q_\theta(s_t, a)$
        Save experience tuple $\{s_t, a_t, r_t, s_{t+1}\}$ to buffer R
        Training θ by gradients
        Sample N tuples $\{s_i, a_i, r_i, s'_i\}$ from replay buffer R (uniformly)
        Compute target $d_j = r_j + \max_{a'} Q_{\theta_j^-}(s'_j, a')$ for $1 \leq j \leq N$
        Compute empirical loss

$$L = \frac{1}{N} \sum_{j=1}^{N} (Q_\theta(s_j, a_j) - d_j)^2$$

        Update $\theta \leftarrow \theta - \alpha \nabla_\theta L$
        Update target network θ^-
        if *counter* mod $\tau = 0$ **then**
            Update target parameter $\theta^- \leftarrow \theta$

There are multiple ways to improve the efficacy of the algorithm.

- **Sampling**: Instead of sampling uniformly from buffer, sample using adaptive distribution (prioritized replay buffer).
- **Double DQN**: DQN overestimates Q values. To mitigate the issue, use Double DQN.
- **Architecture**: Decompose Q value into $Q(s, a) = V(s) + A(s, a)$ and use Dueling DQN architecture.
- **Soft update**: $\theta_{\text{target}} \leftarrow (1 - \beta)\theta_{\text{target}} + \beta\theta$.
- **Exploration constant**: Exploration constant can be made adaptive. Start with large ϵ and gradually decrease ϵ to be small.