Likelihood Machine Learning Seminar II

Lecture 8: Deep Q Learning

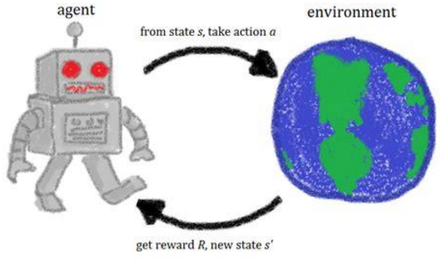
Likelihood Lab

XingYu Fu

1. Markov Decision Process

In this lecture, we are going to train an artificial intelligence *agent* that can interact with the *environment*. Formally, we name this game between the agent and the environment as *Markov Decision Process (MDP)*.

At each stage t of MDP, the environment is at some *state* s_t and the agent receives the state information and chooses an *action* a_t to conduct to the environment. Next, the environment releases an *reward* to the agent as a feedback to agent's response and then changes its own state into s_{t+1}. The process continues until the system triggers some terminal conditions. Below is an intuitive illustration of MDP.



2. Bellman Equation of Q function

Define $Q(s_t, a_t)$ as the quality function describing the value of action a_t in the state s_t. By the theory of discounted MDP, we can prove that the Q function satisfies the so-called *Bellman Equation* or *Optimal Equation*, i.e.

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

, where the max operation is taken over all possible actions the agent can take at the $t + 1$ stage and γ is the discounted factor balance the trade-off between immediate reward and future expected value.

3. Deep Q Learning

Deep q learning algorithm is a technique that trains *neural network to approximate the Q function*. The network receives a tensor that represents the current environment state and outputs the Q value of each possible actions. The network is trained by minimizing the difference between network outputs and the target computed by Bellman Equation through gradient descent. Mechanisms like *epsilon greedy* and *experience replay* are used to further improve the model performances (see the original paper of DeepMind).

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for