# Report of Deep Reinforcement Learning

Jinkun Chen

## Abstract

Last three weeks, I focused on studying the basic knowledge of reinforcement learning and deep reinforcement learning. The basic reinforcement learning conceptions I learned include Malkov Decision Process(MDP),Bellman Equation, Dynamic Programming, Model Free Control and Value Approxitmation. Finally, I read the oringinal paper *Human-Level Control with Deep Reinforcement Learning* of Deepmind Company, and become familiar with DQN algorithm.

## Model Free Control with Easy21 Game

While learning basic knowledge of reinforcement learning, I finished the assignment of David Silver's reinforcement learning courses. The environment is a game named easy21, which is implemented myself. Then I use Monte-Carlo method and Temporal-Difference method(named also Sarsa Control) to train my agent.

The main steps of MC control is to build the episode using $\epsilon$ - Greedy policy. Then, sampling the $k$th episode using $\pi$:$\{S_1, A_1, R_1, ..., S_T\}$ for each state $S_t$ and the action $A_t$ , to update the Q-value function: $Q(S_t, A_t) = Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$ , $N(S_t, A_t) = N(S_t, A_t) + 1$ .

Then I implement the Sarsa($\lambda$) control, specifically the Backward-view Sarsa($\lambda$) control using eligibility traces and $\lambda$ is set to 0. The eligibility traces $E$ is calculated as followed.

$E_0(s, a) = 0 \quad E_t(s, a) = \gamma\lambda E_{t-1}(s, a) + 1(S_t = s, At = a)$

The update method of Sarsa($\lambda$) becomes $Q(s, a) = Q(s, a) + \alpha\delta_t E_t(s, a)$ , $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$

The training result is that both MC and TD control can make the agent win the game almost 2000 times every 10,000 times.

## Deep Reinforcement Learning using DQN

The main idea is to use a neural network to approximate the Q-value funciton, namely Q-network. The network is composed of 3 convex layer and a fully connected layer. The state $S_t$ is obtained from the screen input $x_t$. $S_1$ = $\{x_1\}$ and $S_{t+1} = S_t, a_t, x_t$ . The input of this network is the preprocessed state $\phi_t = \phi(S_t)$. The target to update the network is $y_j = r_j + \gamma max_{a'} Q(\phi_{j+1}, a_j; \theta)$

The first experiment is the replay of original experiment of this paper, using the RMSProp algorithm with minibatches of size 32 and $\epsilon$ annealed linearly from 1 to 0.1 over the first million frames, and fixed 0.1 at thereafter.

Then I try to improve the model by adding a new convolutionary layer with the input size $64 \times 64$ and the output size $128 \times 128$. During the training process, evaluate the model every 50,000 times. However, due to the performance of my computer is not good enough and the training proccess is too low to get the full 50,000,000 iterations. Although I trained the model for almost a week, I only get a part of learning process.

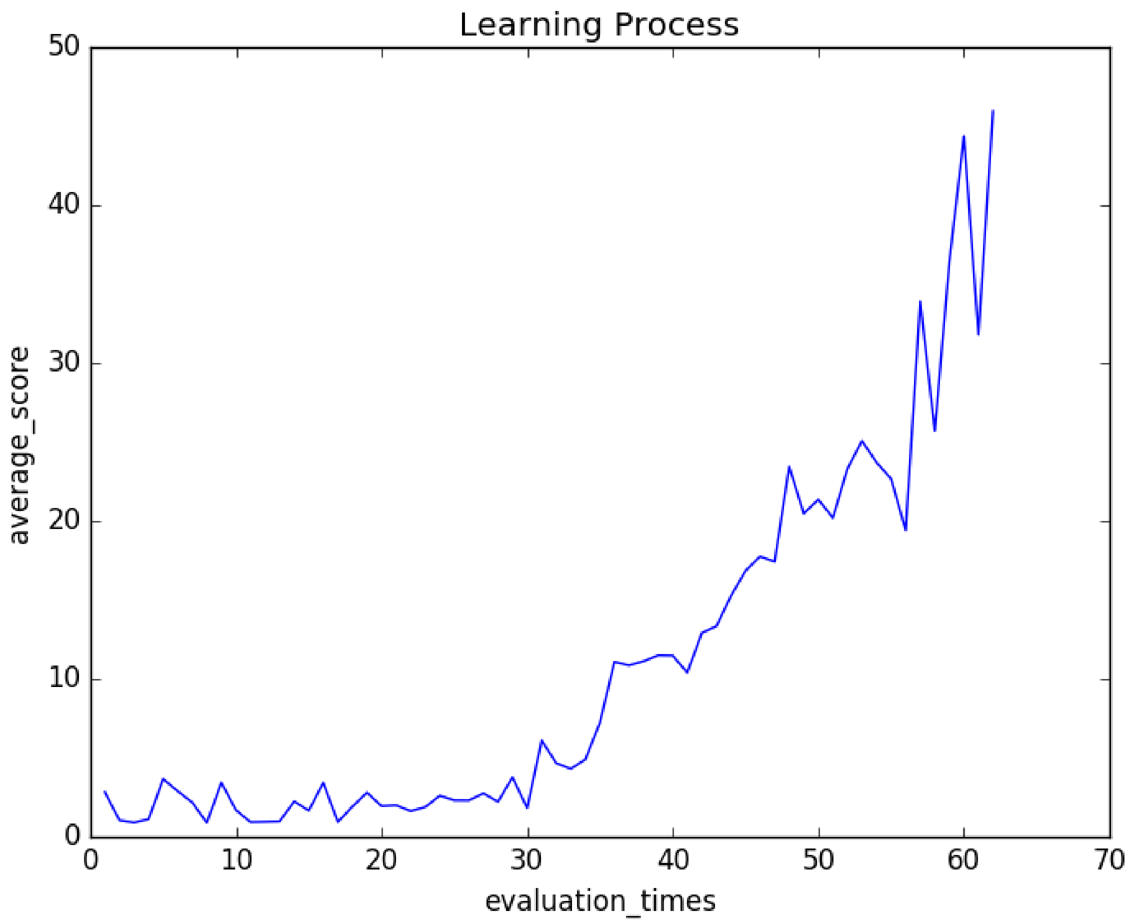

figure 1 The original learning process

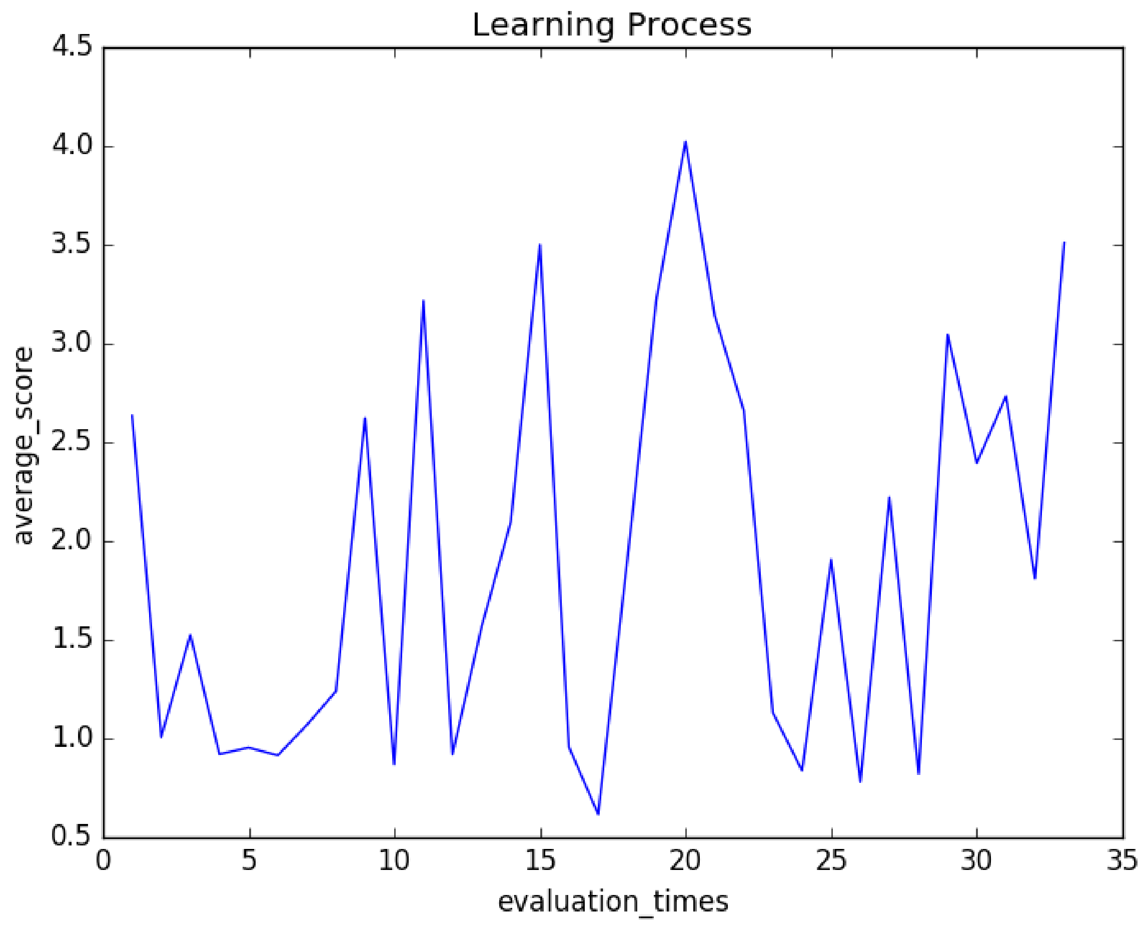

figure 2  The learning process after adding a new layer