Reinforcement learning

introduction

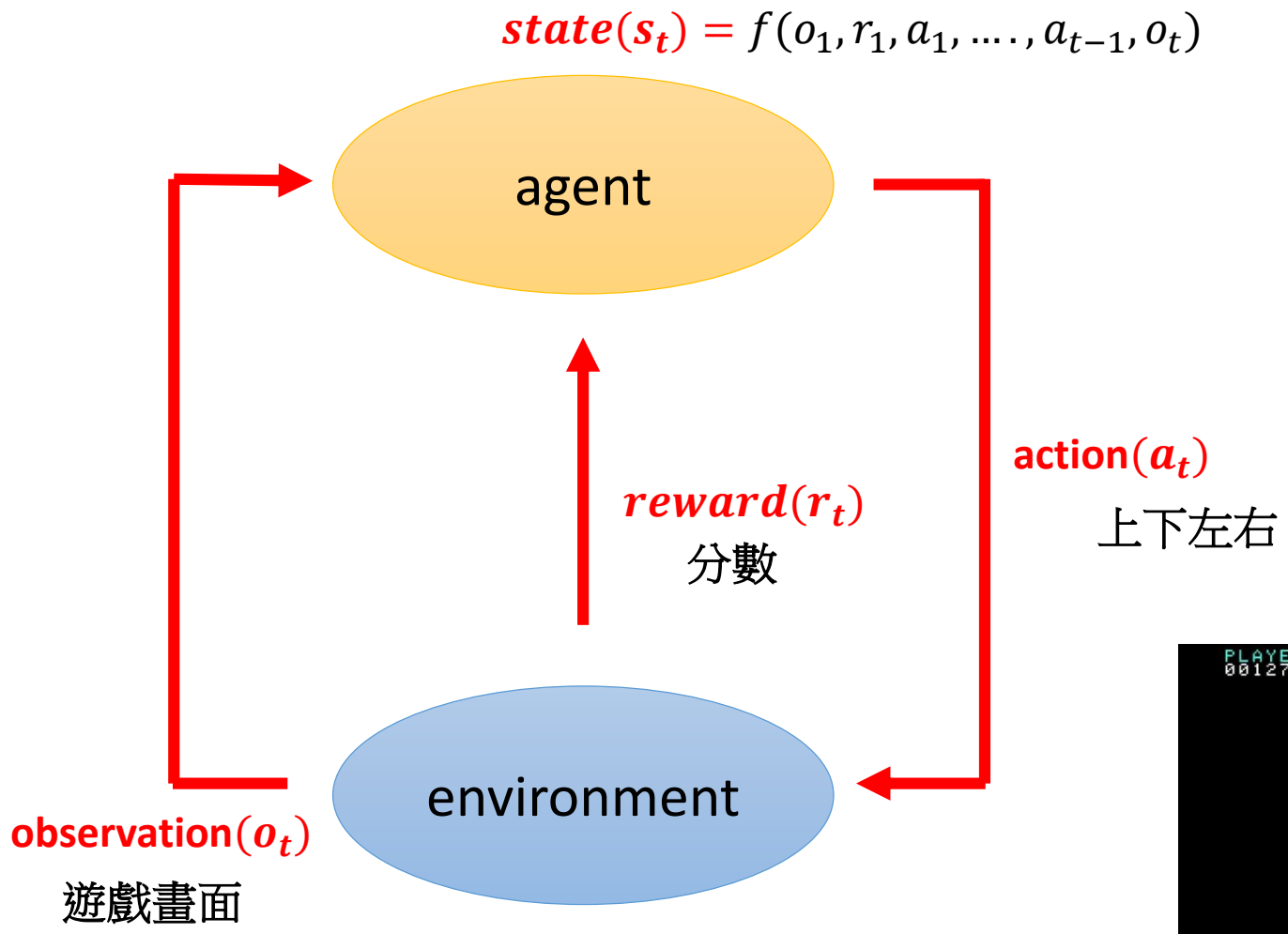
生機四 b06611032 武敬祥

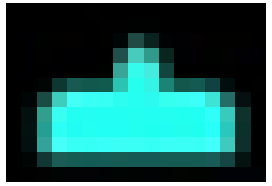
outline

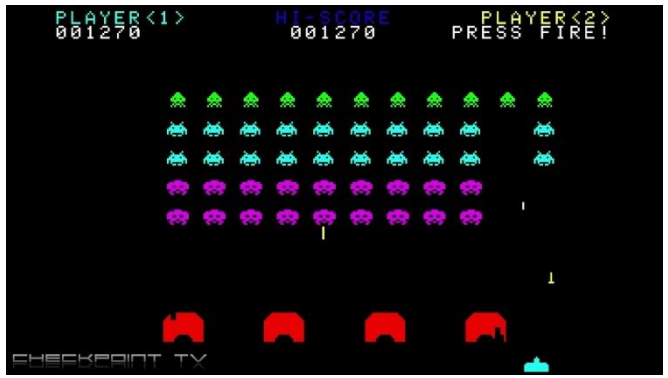
- Reinforcement learning introduce
- Implementation
- Training demo
- Challenge……
- Conclusion

Reinforcement learning introduce

- Structure







Goal : maximize the reward(r_t)

- markov decision process(MDP)

$$S_t = f(o_1, r_1, a_1, \ldots\ldots, a_{t-1}, o_t)$$

$$P(s_{t+1}|S_t) = P(s_{t+1}|S_1 \ldots\ldots\ldots S_t)$$

- Action-value function

$$Q^{\pi}(s,a) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots |s,a]$$

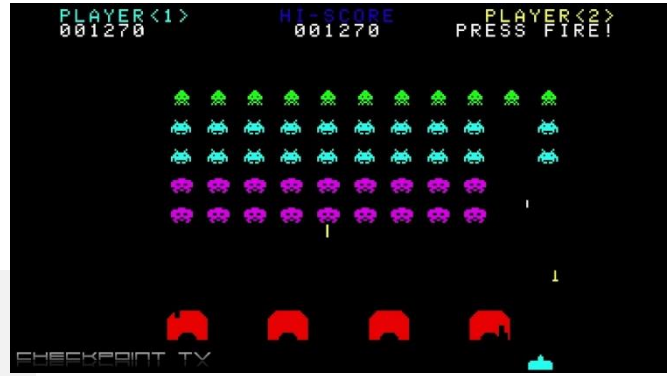
$\gamma = discounting\ factor$

$\pi = policy$

Bellman function

$$Q^*(s,a) = E_{s'}[r_{t+1} + \gamma \max_{a'} Q^*(s',a') \,|s,a]$$

- Q-learning pseudo code



Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S state
    Take action A, observe R, S' Next state
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

真實值(? 估計值

Episode : 一場遊戲

Step of episode : 一場遊戲中的一個動作

→ **Optimization-Temporal difference**

Value iteration :

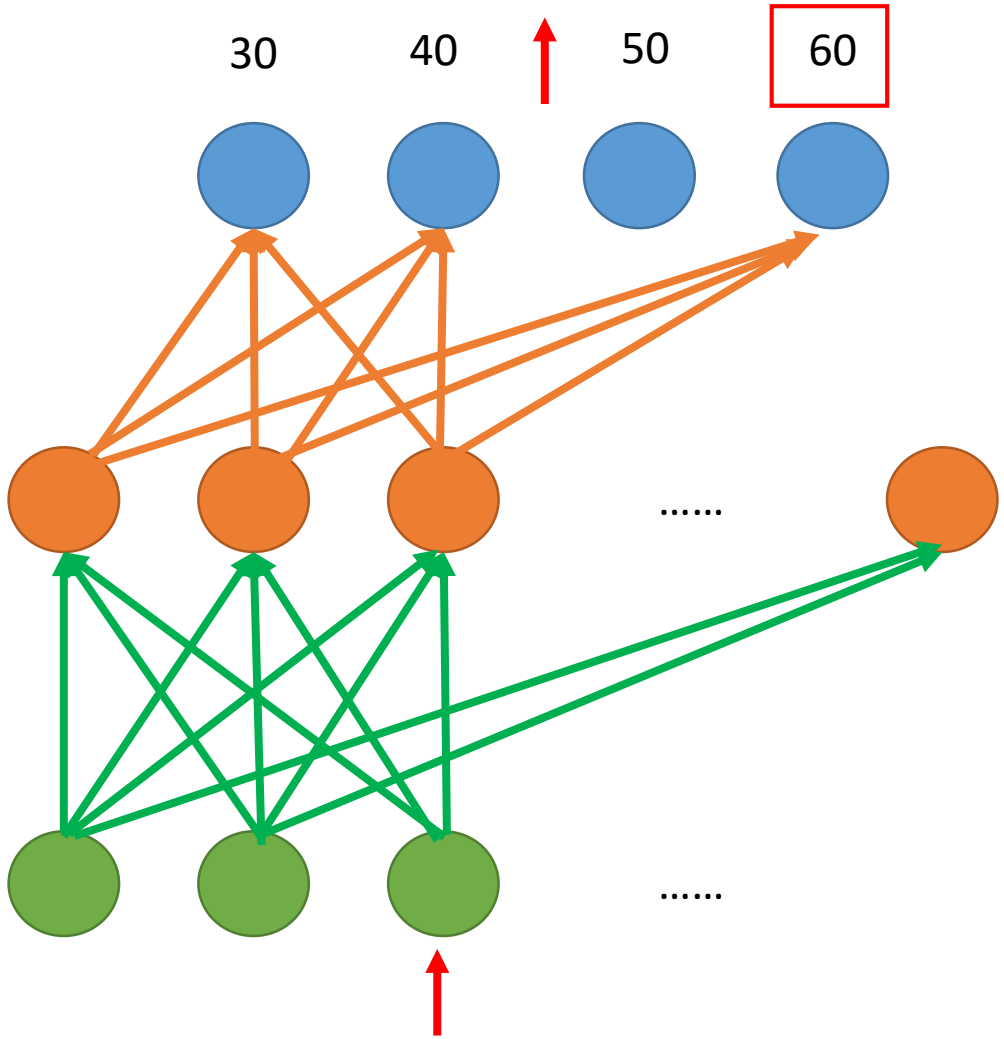
$$Q^{k+1}(s,a) = Q^k(s,a) + \alpha[r_{t+1} + \gamma\, Q(s',a') - Q^k(s,a)]$$

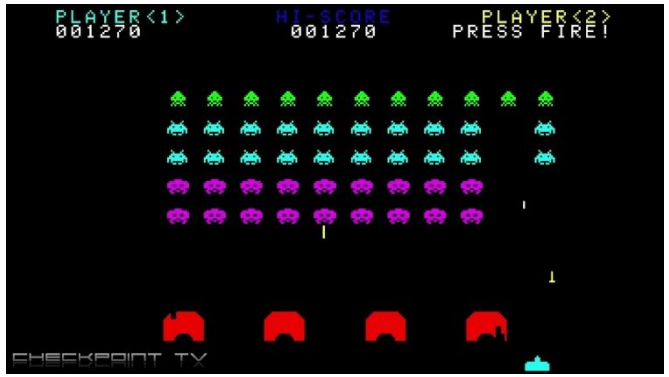
Target net **policy net**

action(a_t) 上(1) 下(2) 左(3) 右(4)

$reward(r_t)$

policy net /Target net

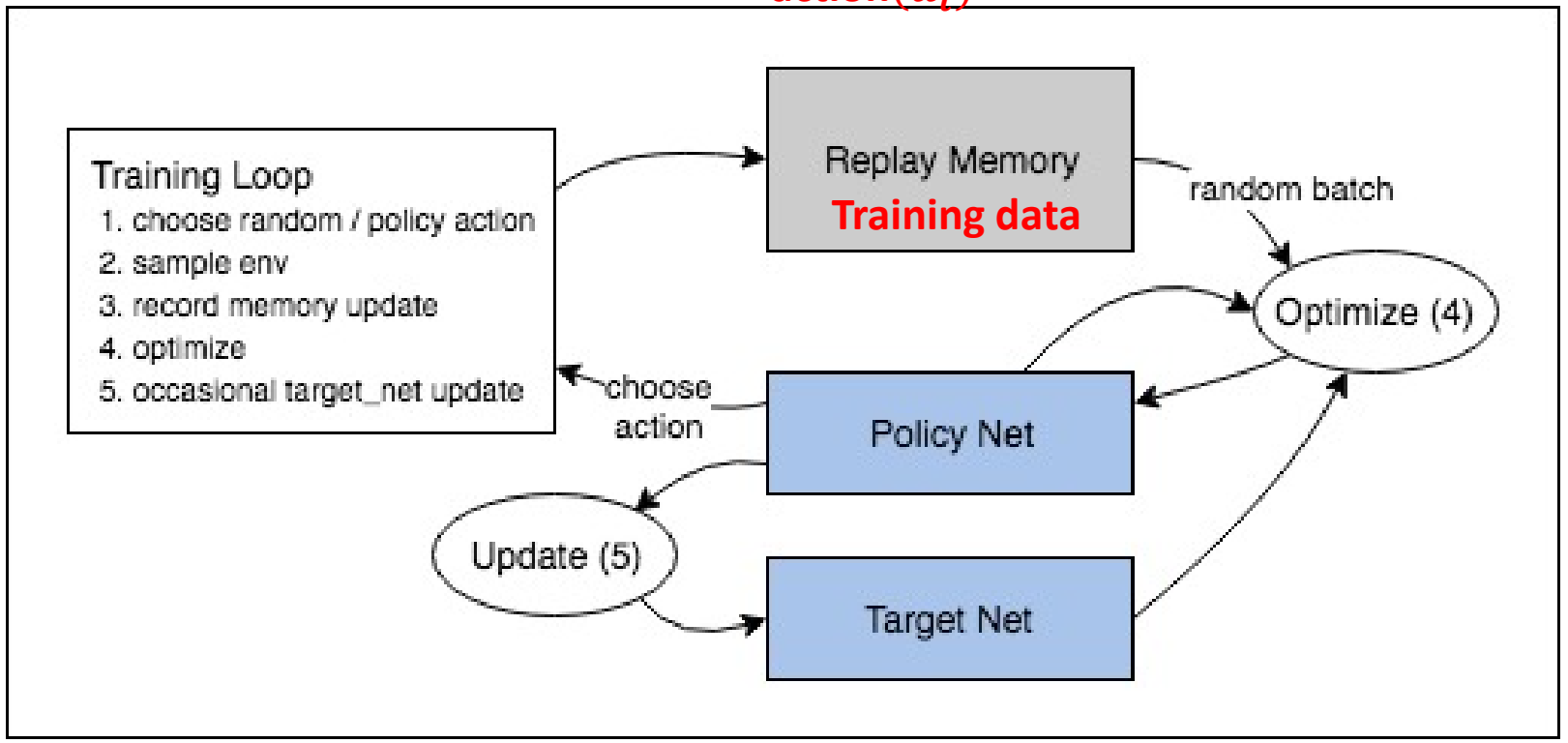




$state(s_t) = f(o_1, r_1, a_1, \ldots, a_{t-1}, o_t)$

Implementation

$reward(r_t)$
observation(o_t)
action(a_t)

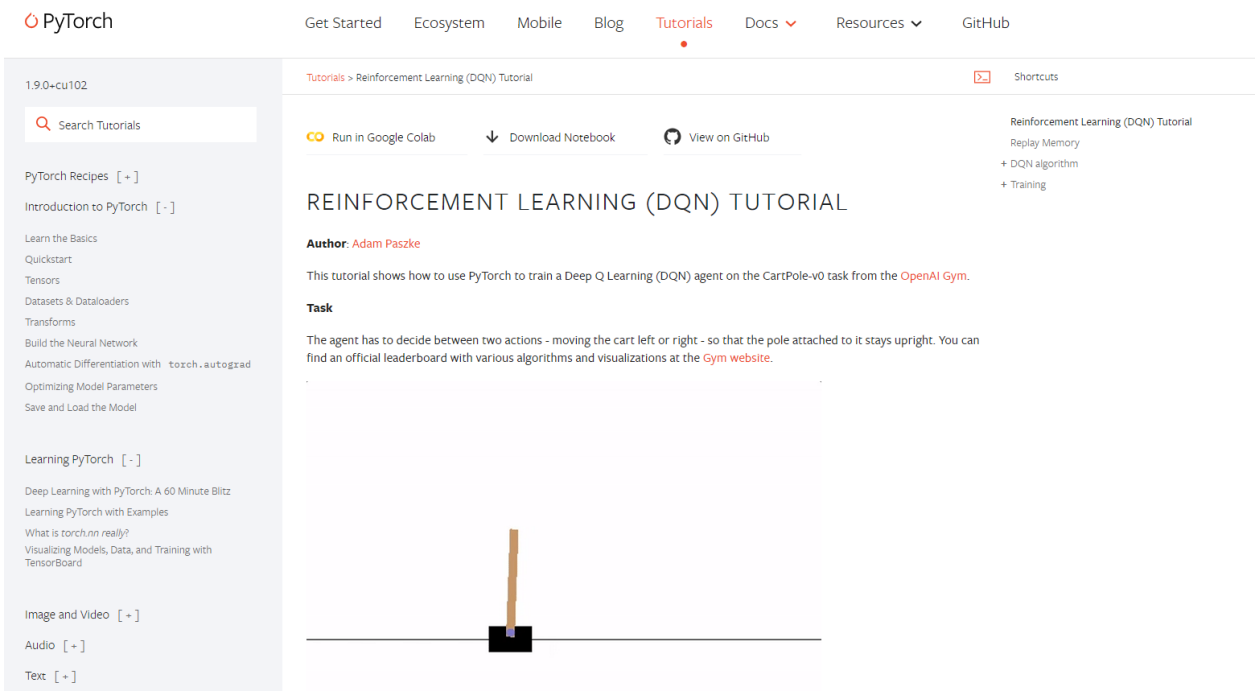


tool

Gym

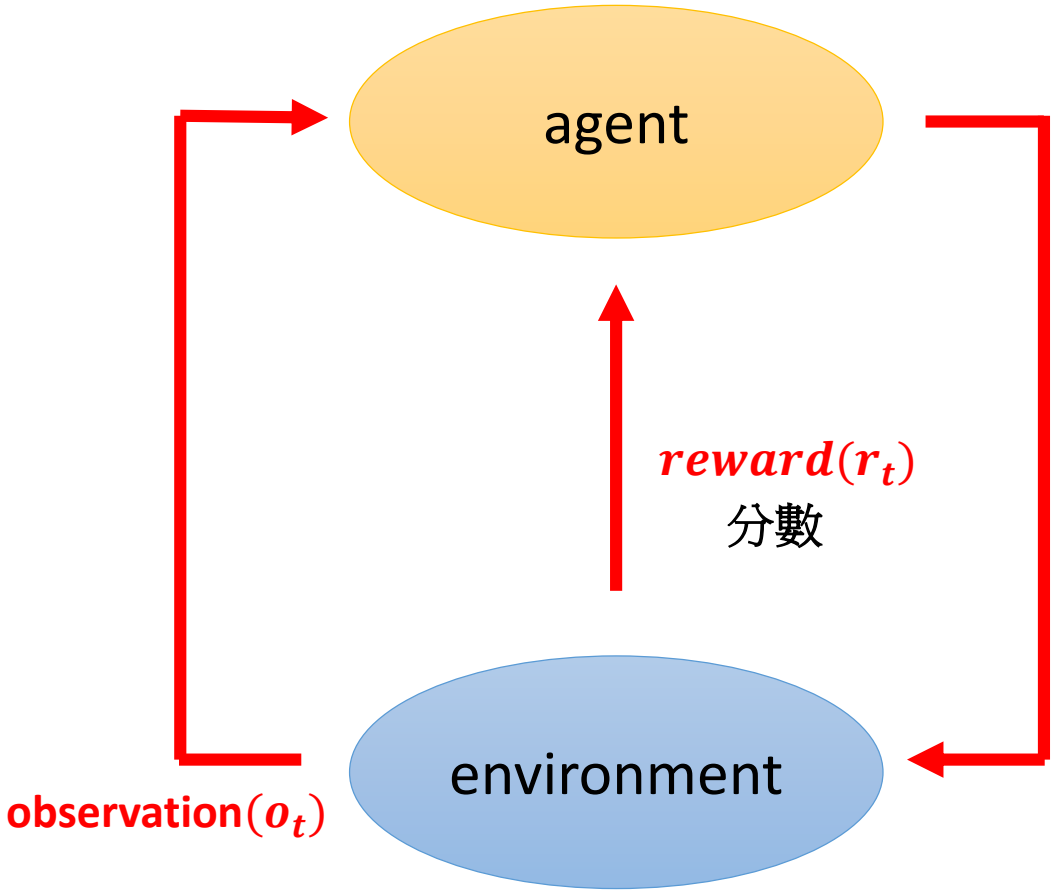
Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

View documentation ›
View on GitHub ›



Implementation -game2

$state(s_t)$ =1000*1000 matrix



player_module.py

遊戲畫面狀態 (P1_code, P1_x, P1_y)

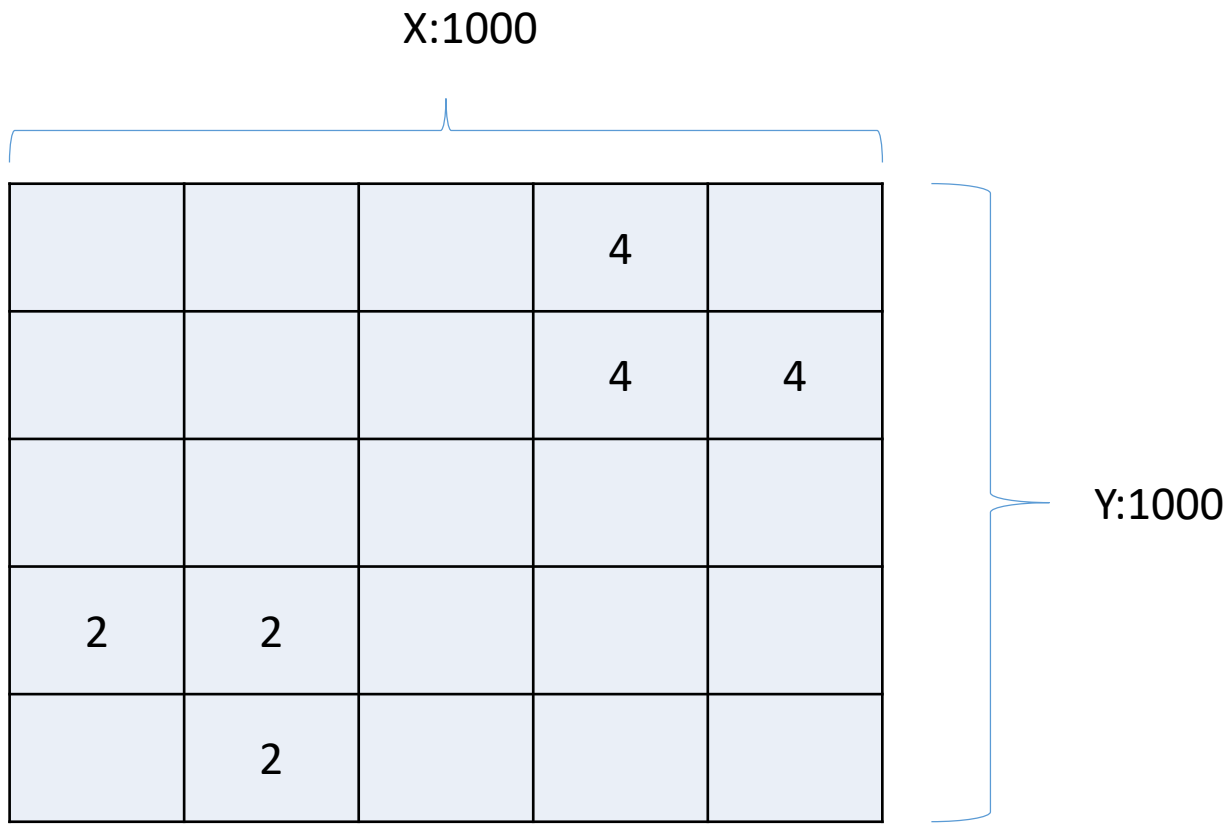
Goal : maximize the reward(r_t)

- $\boldsymbol{reward(r_t)}$ **:** score

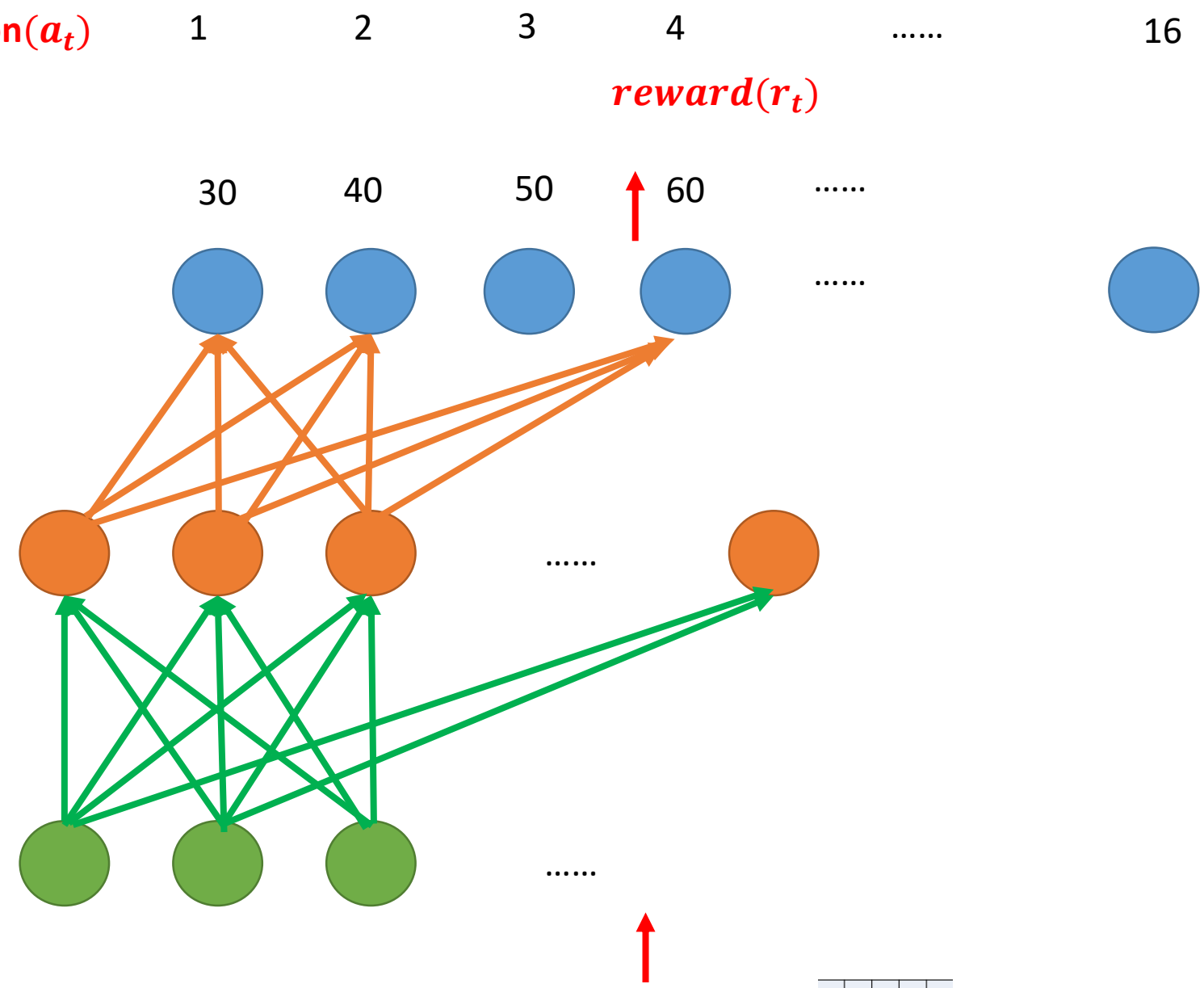
- **action**(a_t) : four state [0 , 1 , 2 , 3]
  probability [0.1,0.2,0.3,0.4]
  [0.1,0.3,0.2,0.4]
  [0.1,0.4,0.2,0.3]
  ⋮

16 combinations

- **observation(o_t)**: (P1_code, P1_x, P1_y)



policy net /Target net



Training demo

Challenge……

- 將 game2.py 設做 environment
- Training time - > difficult to validate the performance
- Action ? Reward? observation?

Conclusion

It works , but we don’t know if it does work

Thanks for listening!!!

Reference

- NTU ADLXMLDS
- Mnih, Volodymyr & Kavukcuoglu, Koray & Silver, David & Graves, Alex & Antonoglou, Ioannis & Wierstra, Daan & Riedmiller, Martin. (2013). Playing Atari with Deep Reinforcement Learning.
- NTU csie machine learning foundation/technique