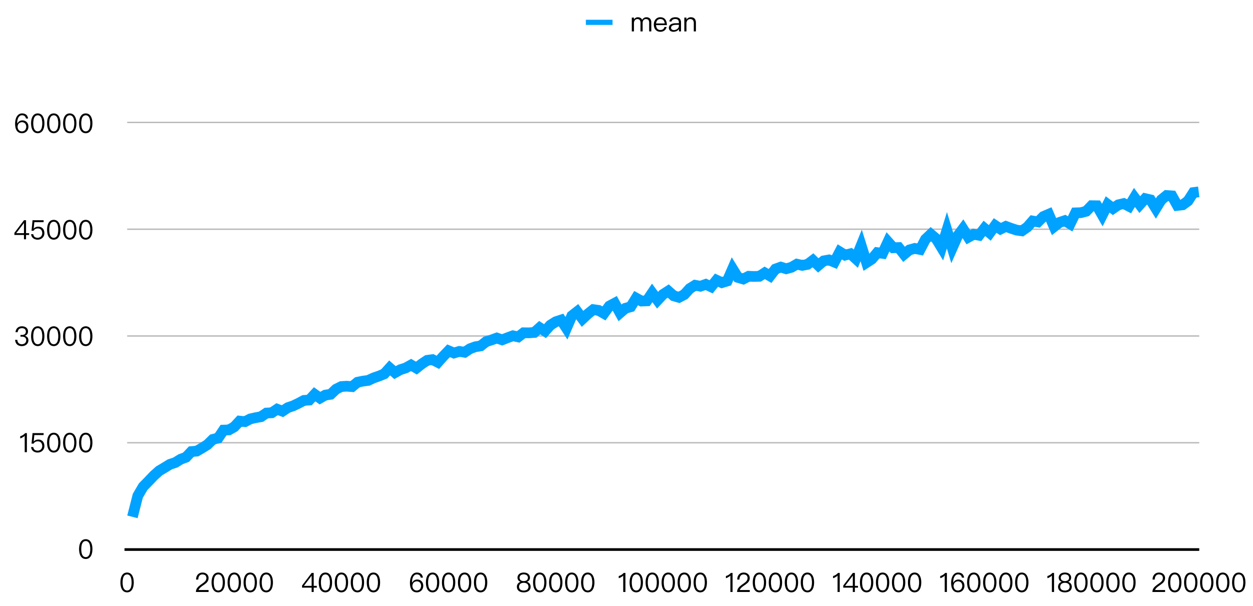
Lab3 2048

Deep Learning and Practice

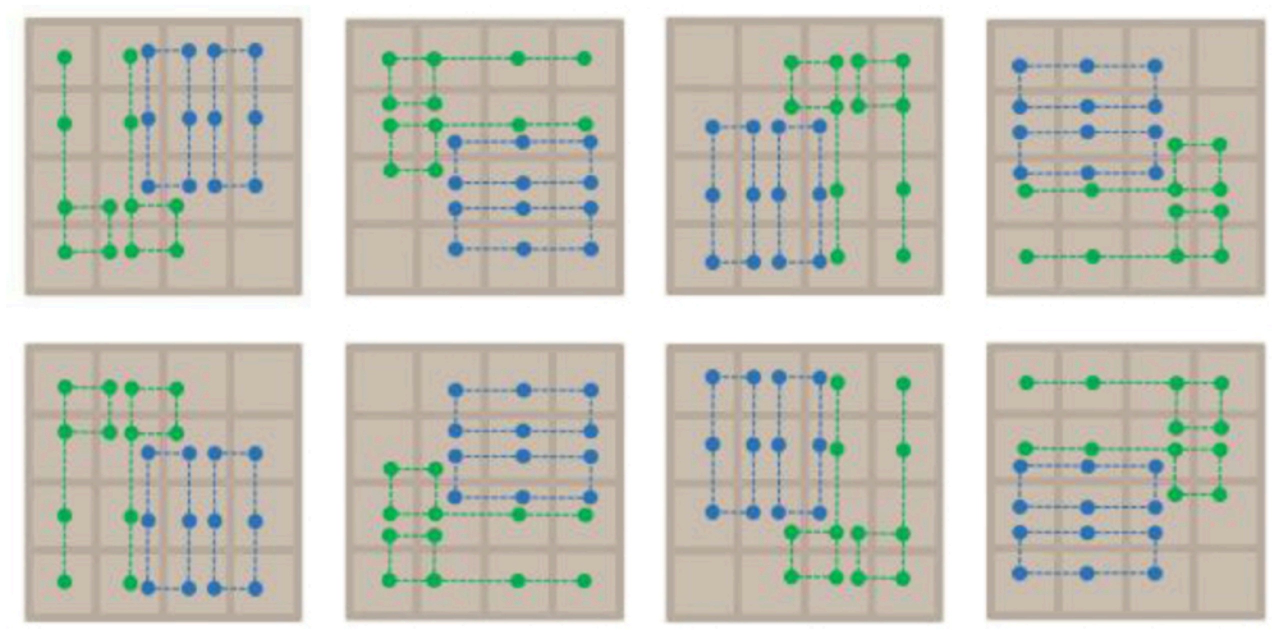
資科工碩1 游騰德 310551054

Episode scores of training



Implementation and the usage of n-tuple network

Since the count of all states are too large for the memory, we select some tuples to describe the state. We can view these tuples as the features of the board state. In this lab, I chose to implement a 4 6-tuple design as below to train the network. By applying transpose and mirror, I am able to use the method to calculate the 8 isomorphisms patterns of the original design pattern.

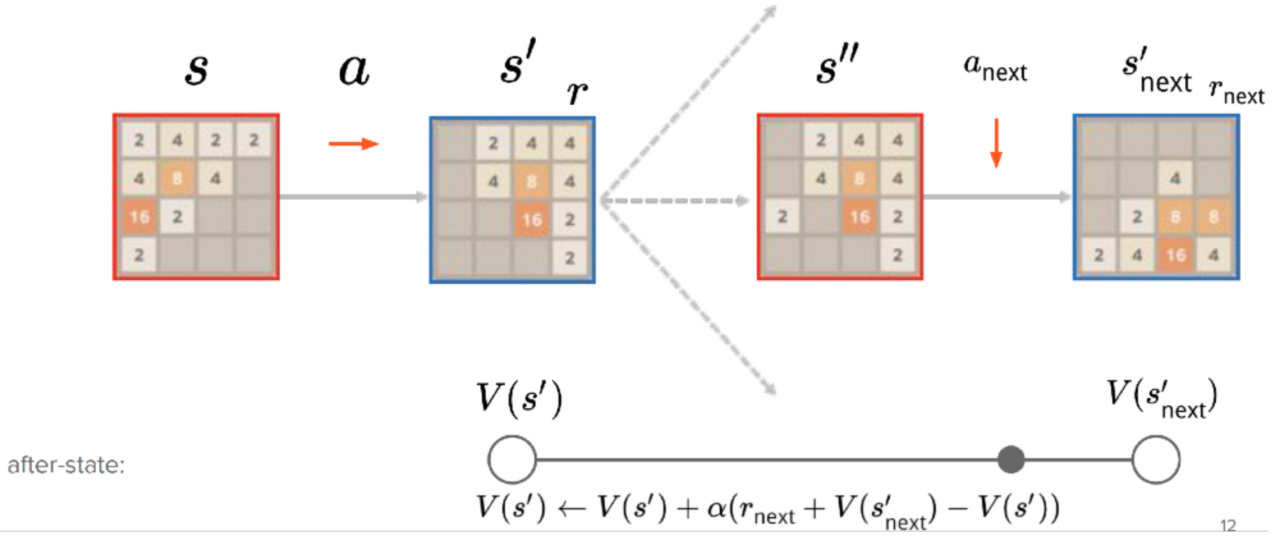


Mechanism of TD(0)

TD(0) is the simplest temporal-difference learning algorithm. While the updating of Monte-Carlo learning requires the terminate of an episode, that of TD learning does not. The feature of TD(0) learning is that it updates the value and learns during each states. Comparing to Monte-Carlo learning, TD(0) is more biased yet has lower variance.

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

TD-backup diagram of V(after-state)



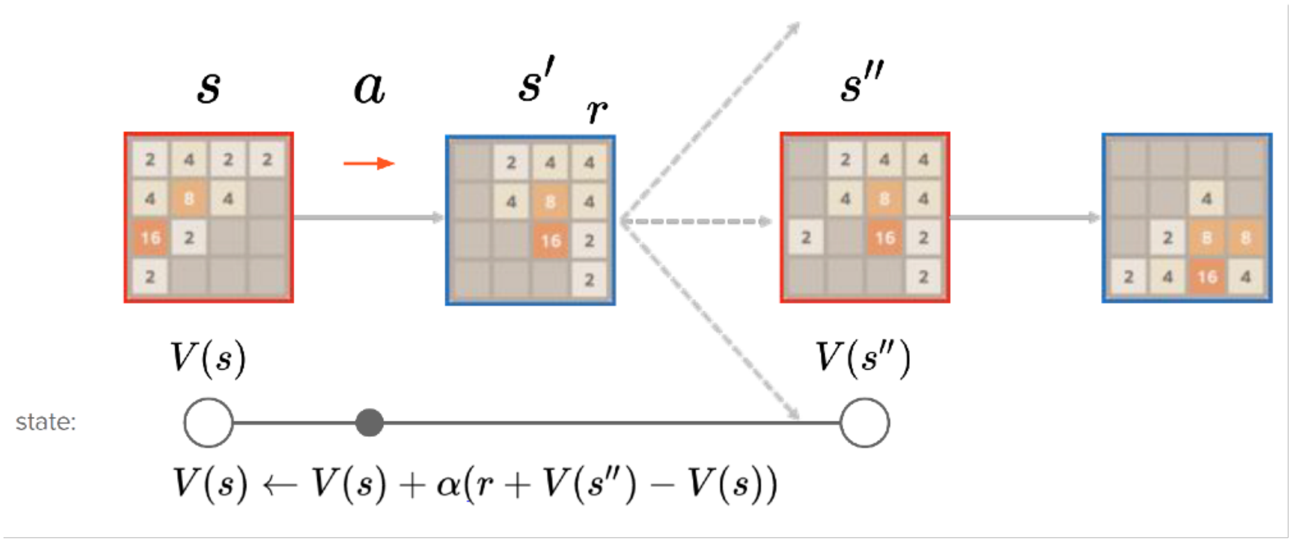
After state **s'** is the state after action **a** at **s**. State **s''** is the state after a random pop up at **s'**. State **s'** next is the state after action **a next** at **s''**. The exact update value for **V(s')** is **r next + V(s' next)**, where **V(s' next)** is the max value of all states' values after different actions at current state.

Action selection of V(after-state) in a diagram

The action decision is based on simply estimating the best value of the states by simulating each available move.

$$\pi(s) = argmax_a(Q(S_t, a))$$

TD-backup diagram of V(state)



After state **s'** is the state after action **a** at **s**. State **s''** is the state after a random pop up at **s'**. The exact update value for **V(s)** is **r + V(s'')**

Action selection of V(state) in a diagram

The action decision is based on the max value of adding the reward value and the expected value of the states by simulating each available move, which is less efficient than that of V(after-state).

$$\pi(s) = argmax_a(R_{t+1} + \mathbb{E}[V(S_{t+1}) \mid S_t = s, A_t = a])$$

Implementation details

There are totally 5 TODOs in the sample codes. 2 of them are virtual functions **estimate(...)** and **update(...)** in class **pattern**. The **estimate** function simply returns the total value of each isomorphisms patterns by looking up the weight table, and the **update** function updates the value of each isomorphisms patterns of a given board, and returns the updated value.

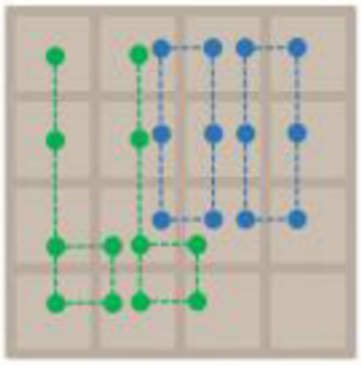
The **indexof(...)** function in class **pattern** returns the given pattern's index of its weight table. Since each implemented pattern in this lab is 6-tuple pattern, the function translate the 6 values of the corresponding positions according to the pattern to a 4 * 6 = 24 bits code.

The **select_best_move(...)** function in class **learning** estimates the values of states by applying all available actions to the given board and return the best action according to the highest estimated value.

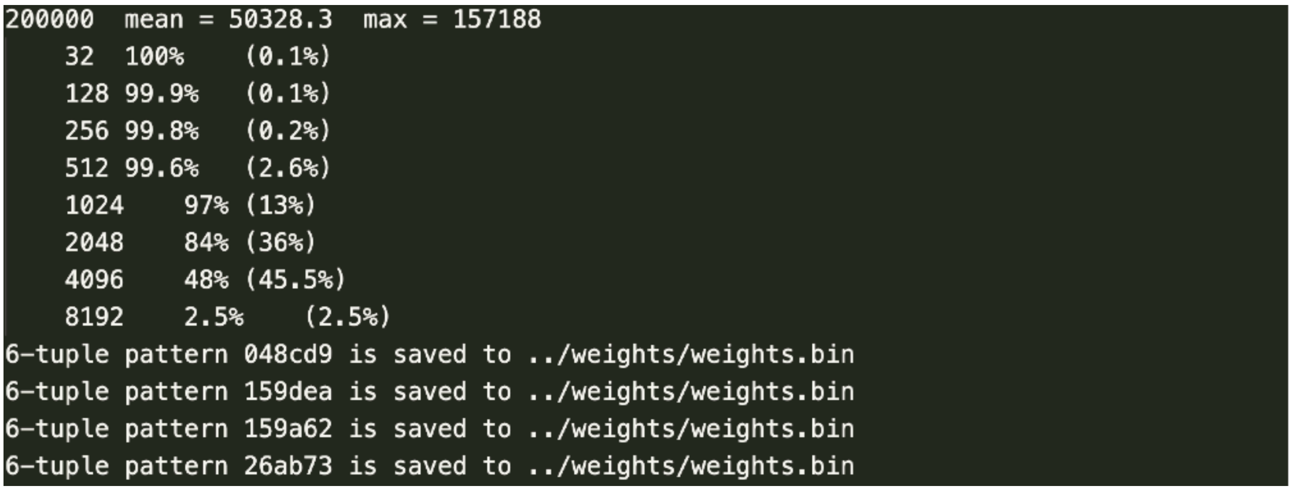
The **update_episode(...)** function in class **learning** updates the tuple network by an episode. The function calculates the TD error of each step of the given path, and call the update function to update all tuple's weight table. The given alpha value act as learning rate in order to control the updating degree during the procedure.

Discussions or improvements

I have tried several different designs of tuples and found out that the design below reaches the highest 2048 rate (84%) at 200,000 episodes. Additionally, I have tried with 4, 8, and no isomorphisms patterns, and the 8 isomorphisms patterns mechanism results in the best performance. Also, I realized that my training process may terminate due to numeric exception. It happened rarely and did not appear again after I decrease the alpha value. Therefore, my guess is that the process occur to some overflow issue under certain unexpected circumstances during updating procedure.



Training results



```
200000  mean = 50328.3  max = 157188
    32  100%    (0.1%)
    128 99.9%   (0.1%)
    256 99.8%   (0.2%)
    512 99.6%   (2.6%)
    1024    97% (13%)
    2048    84% (36%)
    4096    48% (45.5%)
    8192    2.5%    (2.5%)
6-tuple pattern 048cd9 is saved to ../weights/weights.bin
6-tuple pattern 159dea is saved to ../weights/weights.bin
6-tuple pattern 159a62 is saved to ../weights/weights.bin
6-tuple pattern 26ab73 is saved to ../weights/weights.bin
```