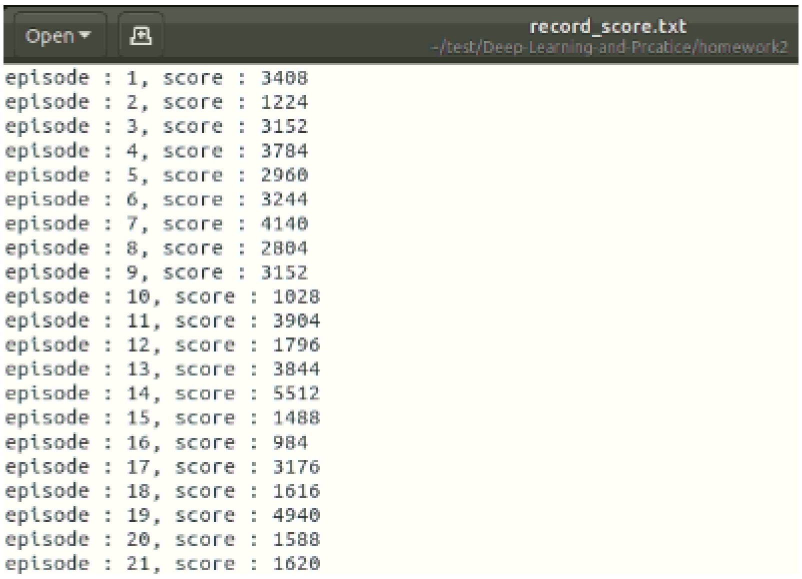
DLP Homework 2

電控碩 0860077 王國倫

A plot shows episode scores

In the training, I record all episode scores, stored into txt file, shown as Fig.1, and visualize the result in Fig.2.



```
episode : 1, score : 3408
episode : 2, score : 1224
episode : 3, score : 3152
episode : 4, score : 3784
episode : 5, score : 2960
episode : 6, score : 3244
episode : 7, score : 4140
episode : 8, score : 2804
episode : 9, score : 3152
episode : 10, score : 1028
episode : 11, score : 3904
episode : 12, score : 1796
episode : 13, score : 3844
episode : 14, score : 5512
episode : 15, score : 1488
episode : 16, score : 984
episode : 17, score : 3176
episode : 18, score : 1616
episode : 19, score : 4940
episode : 20, score : 1588
episode : 21, score : 1620
```

Fig.1 Output result stored in txt file

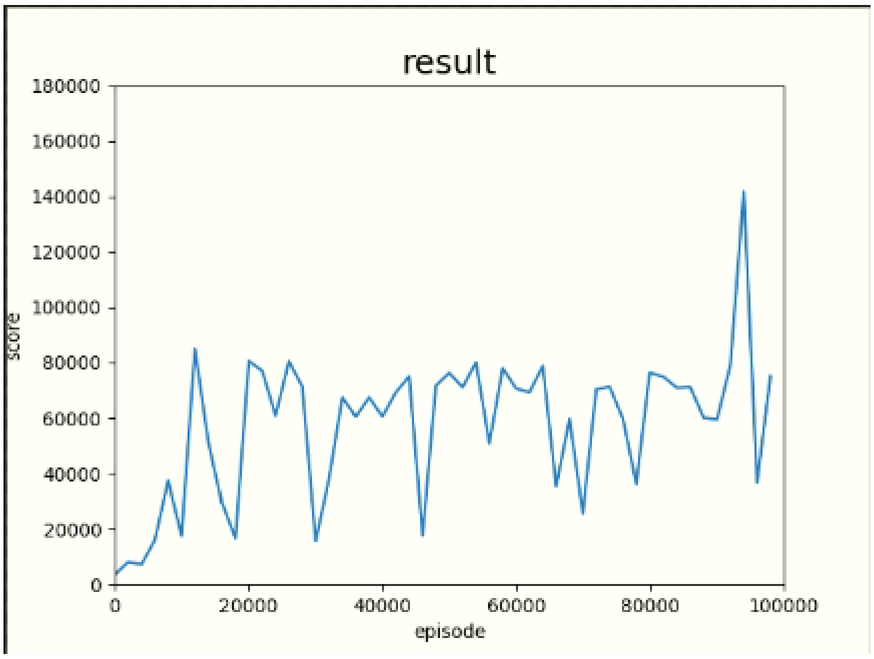


Fig.2 Visual result

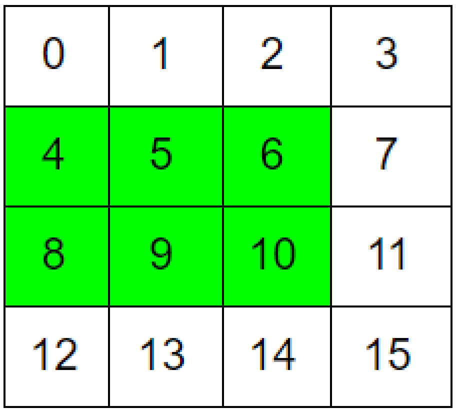
Describe the implementation and the usage of n-tuple network

When calculating the value state function, we must know the value on the board. In order to simplfiy the calculation, the exponent is used as a representative. The largest tile is $2^{15} = 32768$. There are 16 different tiles, but if record all the values on the board, the 2048 game has $(4 * 4)^{16} \approx 1.6 * 10^{19}$ states, the input can be regarded as a one-hot vector, it is impossible to stored each state, so use the n-tuple network method to decrease parameters.

Each n-tuple has 16^n parameters. Compared with recording all the state of boards, this method can use multiple small tuples for calculations as estimated values to save space and easy to calculations.

In order to use the n-tuple network to estimate, you can use the following command to add an n-tuple network, the number is the index of the board, take 6-tuple as an example.(Fig.3)

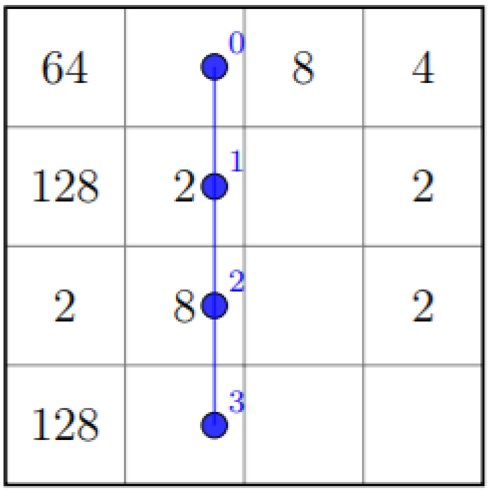
```
tdl.add_feature(new pattern({ 4, 5, 6, 8, 9, 10 }));
```



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Fig.3 6-tuple network

n-tuple network is a large number of input nodes, input value are either 0 or 1, and is a one-hot vector(sparse vector), no hidden layers and only one output, you can see Fig.4. The n-tuple network input value is 0130, so only 0130 is 1, the other is 0, and get it weight.



0123	weight
0000	3.04
0001	−3.90
0002	−2.14
⋮	⋮
0010	5.89
⋮	⋮
0130	**-2.01**
⋮	⋮

Fig.4 the mechanism of n-tuple network

Explain the mechanism of TD(0)

The TD(0) is a special case of TD(λ), and also means it will look ahead one step. Instead of using the accumulated sum of discounted rewards (G_t) we will only look at the immediate reward (R_{t+1}), plus the discount of the estimated value of only 1 step ahead (V(S_{t+1})).

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

In the 2048 game, the agent take a action a, observer a state transition $s \rightarrow s^{''}$, and get a reward r, so the TD(0) formular can shown as below.This rule want to minimize the current prediction and one-step-ahead prediction, and renew this value with learning rate.

$$V(S) \leftarrow V(S) + \alpha(r + V(S^{''}) - V(S))$$

Explain the TD-backup diagram of V(after-state)

The after-state method only carry about after state, this approach no need know environment model, is a model-free method. The TD target is $r_{next} + V(s^{'}_{next})$ according $s^{''}$ and current policy to select next action a_{next}, $s^{''}$ select a_{next} get $s^{'}_{next}$ and the reward r_{next}, then minus V($s^{'}$), calculate TD error. Finally, renew the V($s^{'}$).(Fig.5)

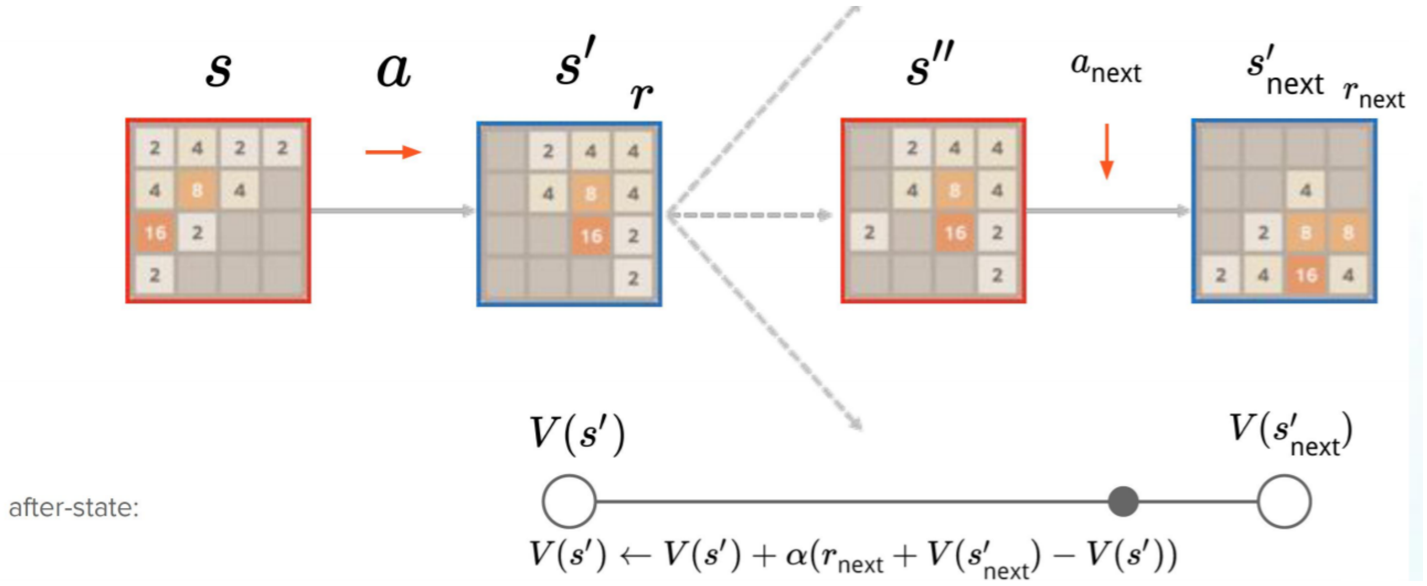


Fig.5 diagram of V(after-state)

Explain the action selection of V(after-state) in a diagram

In order to find the best action, it will calculate take all action, and get next state s' to estimate, then select the highest value to decide next action. (Fig.6)

```
function EVALUATE$(s, a)$
    $s', r \leftarrow$ COMPUTE AFTERSTATE$(s, a)$
    return $r + V(s')$
```

Fig.6 A pseudocode of V(after-state)

Explain the TD-backup diagram of V(state)

The state method is model-base, require environment model to find all next state and transition function, estimate the $V(s^{''})$ and $V(s)$, calculate TD error($r + V(s^{''}) - V(s)$) with learning rate α and then update $V(s)$.(Fig.7)

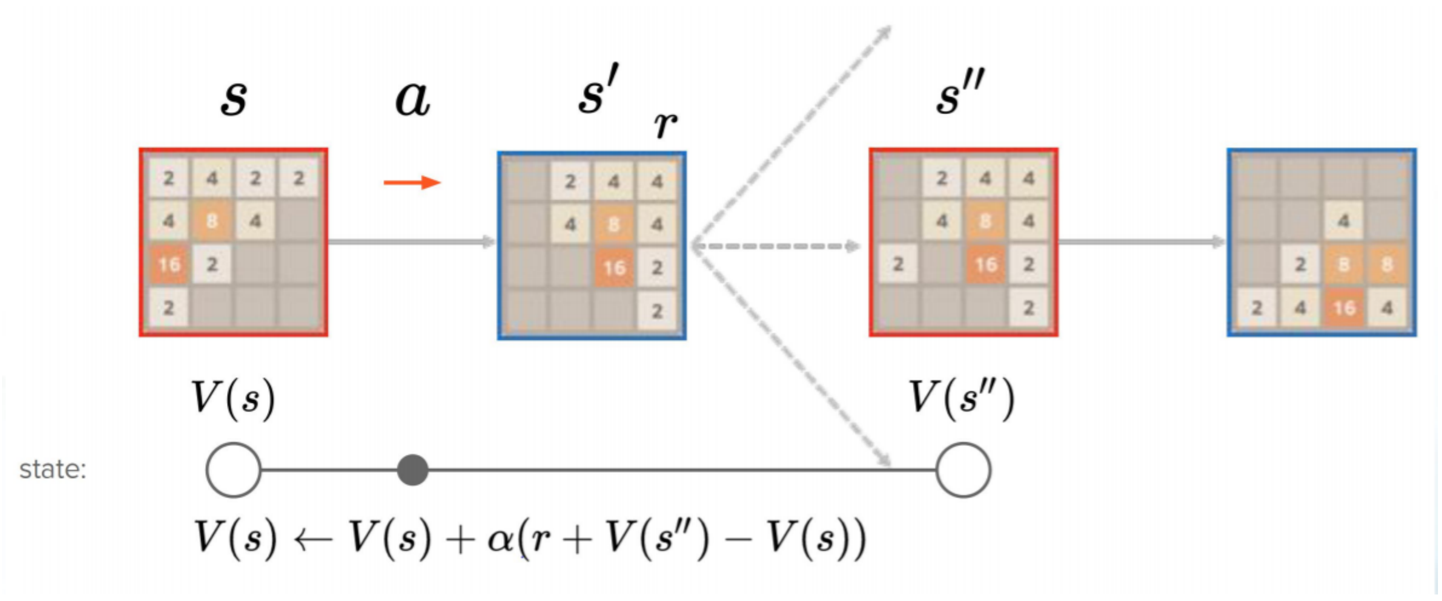


Fig.7 diagram of V(state)

Explain the action selection of V(state) in a diagram

The action selection will take all action, and use next state s' to find all possible next state s'', multiple it transition probability, sum all possible state, then find the highest as next actioin.(Fig.8)

```
function EVALUATE$(s, a)$
    $s', r \leftarrow$ COMPUTE AFTERSTATE$(s, a)$
    $S'' \leftarrow$ ALL POSSIBLE NEXT STATES$(s')$
    return $r + \Sigma_{s'' \in S''} P(s, a, s'') V(s'')$
```

Fig.8 A pseudocode of V(state)

Describe your implementation in detail

In this lab, I change after-state to state, and add some tuples to implement more win rate for 2048 tile, below will introduce I modifiy part.

1. Action selection of V(state), in order to calculate all possible next state, I add two function to find empty position and setting value in specific position.

```
/**
 *  find empty position
 *  return all empty position on the current board
 */
std::vector<int> empty() {
        std::vector<int> result;
        for (int i = 0; i < 16; i++)
            if (at(i) == 0) {
                result.push_back(i);
            }
        return result;
    }
/**
 *  setting value in specific position.
 */
void setting(int position, int value){
        set(position, value);
    }
```

Determining the best action requires all possible next state and it transition probability, the transition function can calculate with environment model, generate 4-tile and 2-tile is 1:9, and divide by the number of empty position.

```
// -----------------------------------
// return reward + sum of all possible move->after_state()
// -----------------------------------

std::vector<int> e = move->after_state().empty();
int mid = e.size();
float val = 0;
for(int i = 0; i < mid; i++){
    board S_next = move->after_state();
    S_next.setting(e[i], 1);
    val += estimate(S_next) * 0.9 / mid;
    S_next = move->after_state();
    S_next.setting(e[i], 2);
    val += estimate(S_next) * 0.1 / mid;
}
move->set_value(move->reward() + val);
```

2. Use 22 tuples to estimate board.

```
tdl.add_feature(new pattern({ 0, 1, 2, 5}));
tdl.add_feature(new pattern({ 0, 1, 2, 3}));
tdl.add_feature(new pattern({ 0, 1, 4, 5}));
tdl.add_feature(new pattern({ 4, 5, 6, 7}));
tdl.add_feature(new pattern({ 1, 2, 5, 6}));
tdl.add_feature(new pattern({ 5, 6, 9, 10}));
tdl.add_feature(new pattern({ 0, 1, 2, 4}));
tdl.add_feature(new pattern({ 0, 1, 2, 6}));
tdl.add_feature(new pattern({ 4, 5, 6, 8}));
tdl.add_feature(new pattern({ 5, 6, 7, 9}));
tdl.add_feature(new pattern({ 0, 4, 5, 6}));
tdl.add_feature(new pattern({ 1, 5, 6, 7}));
tdl.add_feature(new pattern({ 0, 1, 5, 6}));
tdl.add_feature(new pattern({ 1, 2, 6, 7}));
tdl.add_feature(new pattern({ 4, 5, 9, 10}));
tdl.add_feature(new pattern({ 5, 6, 10, 11}));
tdl.add_feature(new pattern({ 4, 5, 6, 9}));
tdl.add_feature(new pattern({ 0, 1, 2, 5, 9}));
tdl.add_feature(new pattern({ 0, 4, 5, 8, 9, 10}));
tdl.add_feature(new pattern({ 1, 5, 6}));
tdl.add_feature(new pattern({ 1, 2, 5, 6, 9}));
tdl.add_feature(new pattern({ 6, 10, 11, 13, 14}));
```

3. Update episode for V(state), the relative formular can refer below. In this lab, the path only save (s,s',a,r), so the s'' is s of next saved move, then just estimate current s and previous s, the we can get TD error, finally, update the V(s).

```
**function** LEARN EVALUATION$(s, a, r, s', s'')$
    $V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$
```

```
void update_episode(std::vector<state>& path, float alpha = 0.1) const {
        path.pop_back();
        state& move_final = path.back();
        float exact = 0 - estimate(move_final.before_state());
        for (path.pop_back() /* terminal state */; path.size(); path.pop_back()) {
            state& move = path.back();
            float error = exact - estimate(move.before_state()) + move.reward();
            exact = update(move.before_state(), alpha * error);
        }
    }
```

Other discussions or improvements

In order to record episode scores, I write a function to save it, and modifiy select best move, because the default move is up, if other result is equal, as a result, I add random move let it can more generalization.

```
void save_score(size_t episode, int score){
    std::ofstream file;
    file.open("./record_score.txt",std::ios::app);
    file << "episode : " << episode << ", score : " << score <<std::endl;
}


//in select_best_move
state* best = after + 4 * rand() % 4;
```

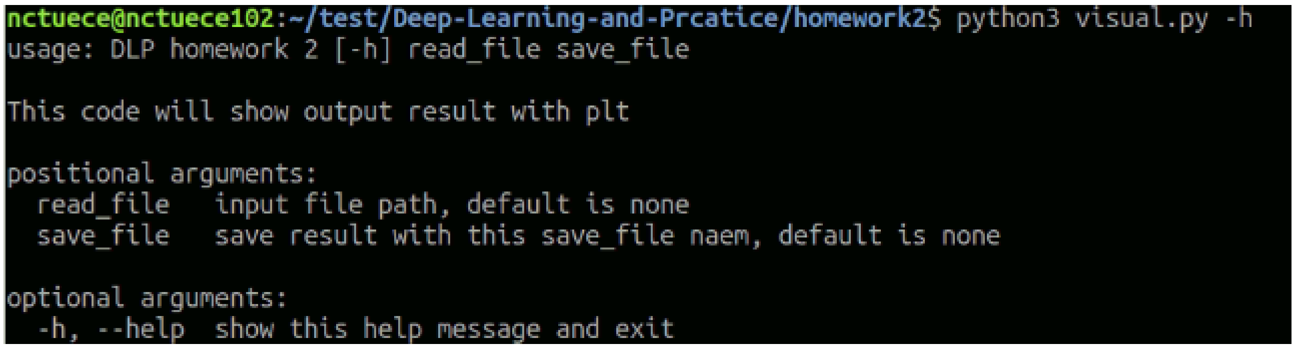
I also adjust learning rate to learning well, because I found it win rate of 2048 always oscillation around 95%, decrease learning rate to 0.01 improve win rate of 2048.

- learning rate 0.1 -> 0.01

The visual code with python shown as below, you can see Fig.9 teach how to use this program, you need to input your recore file path and the save file name. The plt_result will plot 0~100,000 data and per 2,000 will plot, the result can see Fig.2.

```
#!/usr/bin/env python3

import numpy as np
import argparse
import matplotlib.pyplot as plt

def read(file):
    episode = []
    score = []
    with open(file) as f:
        Lines = f.readlines()
        for line in Lines:
            ep, sc = line.split(",")
            episode.append((int)(ep.split(":")[1]))
            score.append((int)(sc.split(":")[1]))

    return np.array(episode), np.array(score)

def plt_result(epoches, score, file):

    plt.title("result", fontsize = 18)

    plt.plot(episode[0:100000:2000], score[0:100000:2000])

    plt.xlim(0,100000)
    plt.ylim(0,180000)

    plt.xlabel("episode")
    plt.ylabel("score")

    plt.savefig('{0}.png'.format(file))
    plt.show()
```

```python
def create_argparse():
    parser = argparse.ArgumentParser(prog="DLP homework 2", description='This code wil

    parser.add_argument("read_file", type=str, default="none", help="read file and sho
    parser.add_argument("save_file", type=str, default="none", help="save result in th

    return parser

if __name__ == "__main__":

    parser = create_argparse()
    args = parser.parse_args()

    episode, score = read(args.read_file)
    plt_result(episode, score, args.save_file)
```



```
nctuece@nctuece102:~/test/Deep-Learning-and-Prcatice/homework2$ python3 visual.py -h
usage: DLP homework 2 [-h] read_file save_file

This code will show output result with plt

positional arguments:
  read_file   input file path, default is none
  save_file   save result with this save_file naem, default is none

optional arguments:
  -h, --help  show this help message and exit
```

Fig.9 Program description

tags: **`DLP2021`**