

## Lab2: Temporal Difference Learning

### Lab Objective:

In this lab, you will learn temporal difference learning (TD) algorithm by solving the 2048 game using an  $n$ -tuple network.

### Important Date:

1. Experiment Report Submission Deadline: 4/13 (Thu) 12:00

2. Demo date: 4/13 (Thu)

### Turn in:

1. Experiment report (.pdf)

2. Source code [NOT including model weights]

Notice: zip all files with name “DLP\_LAB2\_StudentId\_Name.zip”,

e.g.: 「DLP\_LAB2\_0856738\_鄭紹雄.zip」

### Lab Description:

- Understand the concept of (before-)state and after-state.
- Learn to construct and design an  $n$ -tuple network.
- Understand TD algorithm.
- Understand Q-learning network training.

### Requirements:

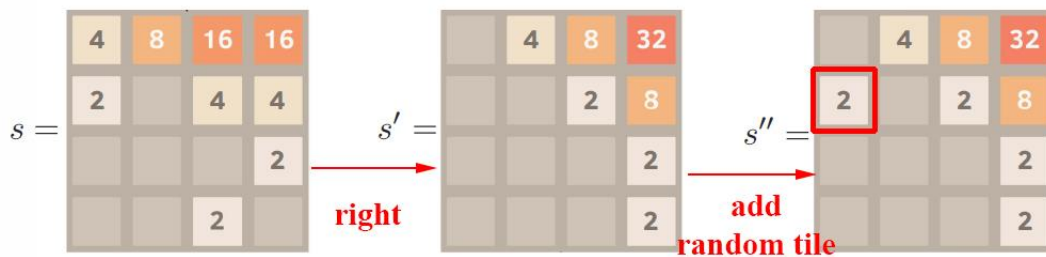
- Implement TD(0) algorithm
  - Construct an  $n$ -tuple network
  - Action selection according to the  $n$ -tuple network
  - Calculate TD-target and TD-error
  - Update  $V(\text{state})$ , **not**  $V(\text{after-state})$ .
  - Understand temporal difference learning mechanisms

Game Environment – 2048:

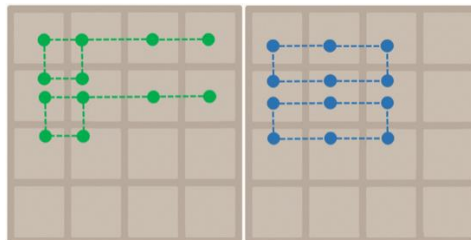
- Introduction: 2048 is a single-player sliding block puzzle game. The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048.
- Actions: **Up, Down, Left, Right**
- Reward: The score is the value of new tile when two tiles are combined.

		2	4
		4	8
	2	16	32
	2	2	16

- A sample of two-step state transition

Implementation Details:**Network Architecture**

- $n$ -tuple patterns:  $4 \times 6$ -tuples with all possible isomorphisms

**Training Arguments**

- Learning rate: 0.1
  - Learning rate for features of  $n$ -tuple network with  $m$  features:  $0.1 \div m$
- Train the network 500k  $\sim$  1M episodes

Algorithm:**A pseudocode of the game engine and training.** (modified backward training method)

```

function PLAY GAME
   $score \leftarrow 0$ 
   $s \leftarrow \text{INITIALIZE GAME STATE}$ 
  while IS NOT TERMINAL STATE( $s$ ) do
     $a \leftarrow \underset{a' \in A(s)}{\text{argmax}} \text{EVALUATE}(s, a')$ 
     $r, s', s'' \leftarrow \text{MAKE MOVE}(s, a)$ 
    SAVE RECORD( $s, a, r, s', s''$ )
     $score \leftarrow score + r$ 
     $s \leftarrow s''$ 
  for ( $s, a, r, s', s''$ ) FROM TERMINAL DOWNT0 INITIAL do
    LEARN EVALUATION( $s, a, r, s', s''$ )
  return  $score$ 

function MAKE MOVE( $s, a$ )
   $s', r \leftarrow \text{COMPUTE AFTERSTATE}(s, a)$ 
   $s'' \leftarrow \text{ADD RANDOM TILE}(s')$ 
  return ( $r, s', s''$ )

```

**TD-state**

```

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

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

```

**TD-after-state**

```

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

function LEARN EVALUATION( $s, a, r, s', s''$ )
   $a_{next} \leftarrow \underset{a' \in A(s'')}{\text{argmax}} \text{EVALUATE}(s'', a')$ 
   $s'_{next}, r_{next} \leftarrow \text{COMPUTE AFTERSTATE}(s'', a_{next})$ 
   $V(s') \leftarrow V(s') + \alpha(r_{next} + V(s'_{next}) - V(s'))$ 

```

### Rule of Thumb:

- You can design your own  $n$ -tuple network, but do NOT try CNN.
- 2048-tile should appear within 10,000 episodes.

### Scoring Criteria:

Show your work, otherwise no credit will be granted.

- Report (60%)
  - A plot shows episode scores of at least 100,000 training episodes (10%)
  - Describe the implementation and the usage of  $n$ -tuple network. (10%)
  - Explain the mechanism of TD(0). (5%)
  - Explain the TD-backup diagram of  $V(\text{after-state})$ . (5%)
  - Explain the action selection of  $V(\text{after-state})$  in a diagram. (5%)
  - Explain the TD-backup diagram of  $V(\text{state})$ . (5%)
  - Explain the action selection of  $V(\text{state})$  in a diagram. (5%)
  - Describe your implementation in detail. (10%)
  - Other discussions or improvements. (5%)
- Demo Performance (40%)
  - The 2048-tile win rate in 1000 games,  $[\text{winrate}_{2048}]$ . (20%)
  - Questions. (20%)

### References:

- [1] Szubert, Marcin, and Wojciech Jaśkowski. "Temporal difference learning of N-tuple networks for the game 2048." 2014 IEEE Conference on Computational Intelligence and Games. IEEE, 2014.
- [2] Kun-Hao Yeh, I-Chen Wu, Chu-Hsuan Hsueh, Chia-Chuan Chang, Chao-Chin Liang, and Han Chiang, Multi-Stage Temporal Difference Learning for 2048-like Games, accepted by IEEE Transactions on Computational Intelligence and AI in Games (SCI), doi: 10.1109/TCIAIG.2016.2593710, 2016.
- [3] Oka, Kazuto, and Kiminori Matsuzaki. "Systematic selection of n-tuple networks for 2048." International Conference on Computers and Games. Springer International Publishing, 2016.
- [4] moporgic. "Basic implementation of 2048 in Python." Retrieved from Github: <https://github.com/moporgic/2048-Demo-Python>.
- [5] moporgic. "Temporal Difference Learning for Game 2048 (Demo)." Retrieved from Github: <https://github.com/moporgic/TDL2048-Demo>.
- [6] lukewayne123. "2048-Framework" Retrieved from Github: <https://github.com/lukewayne123/2048-Framework>.