Lab7: Temporal Difference Learning

Lab Objective:

In this project, you are going to build a bot to play 2048 through reinforcement learning, TD(0). This bot should be able to improve its performance on 2048 through played game sequences (experiences).

Important Date:

1. Experiment Report Submission Deadline: 5/30 (Thu) 12:00
2. Demo date: 6/13 (Thu) or TBD

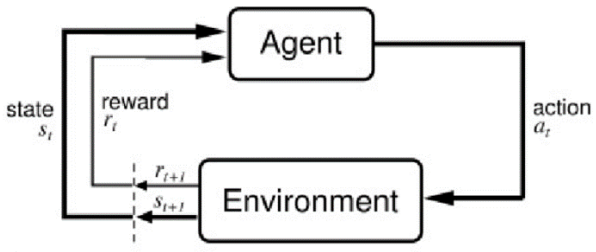
Turn in:

1. Experiment Report (.pdf)
2. Source code [NOT includes your weight]

Notice: zip all files in one file and name it like「DLP_LAB7_your studentID_name.zip」, ex: 「DLP_LAB7_0586036_何國豪.zip」

Lab Description:

- Reinforcement Learning is a computational approach to learning from interaction and is focus on goal-directed learning.
- Agent-Environment Interaction Framework
  - Agent: The learner and decision-maker.
  - Environment: The thing it interacts with, comprising everything outside the agent.
  - State: whatever information is available to the agent.
  - Reward: single numbers.



- Temporal Difference Learning(TD-Learning) is a kind of reinforcement learning and is able to adjust weights automatically.
  - The goal of TD-Learning is to predict the actual return R_t
  - The way it adjusts the weight is through: $V(s_t) = V(s_t) + \alpha(R_t - V(s_t))$
  - **TD(0):** $\boldsymbol{V(s_t) = V(s_t) + \alpha(r_{t+1} + V(s_{t+1}) - V(s_t))}$

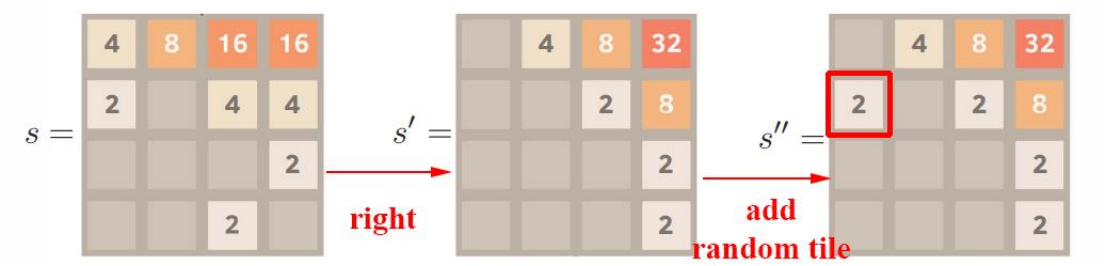
Requirements:

- Understand how TD(0) works.
- Implement TD(0) training algorithm.
- Train 2048 bot with V(state) or V(after-state).
  - If you use C/C++ version, you MUST implement V(state) case

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.

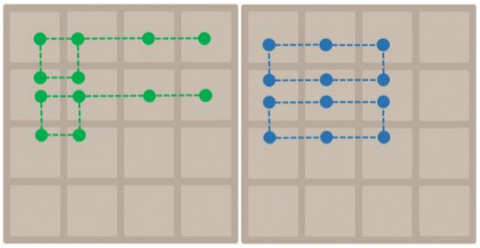
- A sample of two-step state transition



Implementation Details:

Network Architecture

- n-Tuples Pattern: 4 × 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 ~ 1M episodes

Methodology:

A pseudocode of a game engine and training (modified backward training method)

function PLAY GAME
  $score \leftarrow 0$
  $s \leftarrow$ INITIALIZE GAME STATE
  while IS NOT TERMINAL STATE(s) **do**
    $a \leftarrow \underset{a' \in A(s)}{\text{argmax}}$ EVALUATE(s, a')
    $r, s', s'' \leftarrow$ 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 DOWNTO INITIAL **do**
    LEARN EVALUATION(s, a, r, s', s'')
  return $score$

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

TD(0)-state

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

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

TD(0)-afterstate

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

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

Rule of Thumb:

- Try n-tuple network first.
- You may want to try other networks (for bonus), but do NOT try CNN.
- 2048-tile should appear within 10,000 episodes.
- Python [6] is slow (train 100,000 about 2 days). Use C/C++ [5] if you are familiar with them.

Scoring Criteria:

- Report (70%)
  - A plot shows episode scores of at least 100,000 training episodes (10%)
  - Explain the mechanism of TD(0) (10%)
  - Describe how to train and use a V(state) network (10%)
  - Describe how to train and use a V(after-state) network (10%)
  - Describe how the code work (the whole code) (20%)
  - More you want to say (10%)
    - Other study or improvement for the project.
- Strength (30%)
  - {Python version} The 1024-tile win rate in 1000 games, $\lceil \text{winrate}_{1024} \rceil$
  - {C/C++ version} The 2048-tile win rate in 1000 games, $\lceil \text{winrate}_{2048} \rceil$

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