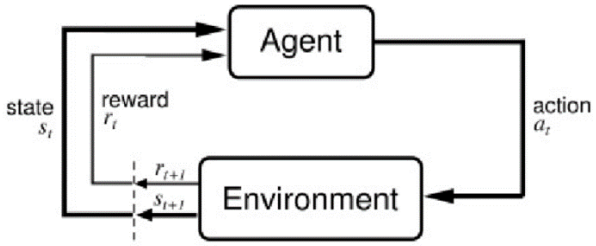
Lab 8: Temporal Difference Learning

Lab Objective:

In this project, you are going to build an AI to play 2048 through reinforcement learning, TD(0). This AI should be able to improve its performance on 2048 through played game sequences/experiences. The output of the AI would be the $Q(s, a)$ of the input state s.

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.



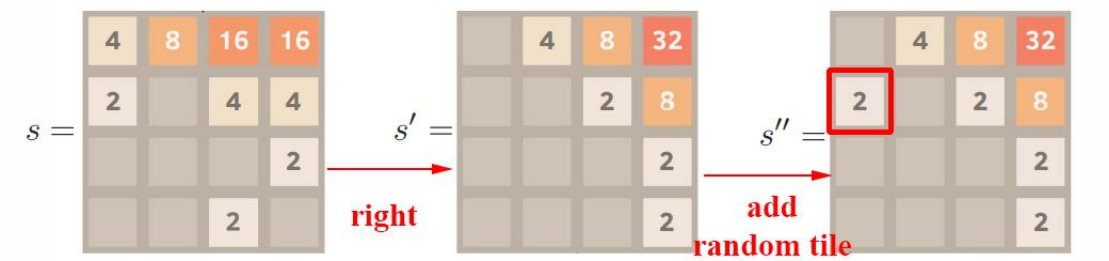
- Temporal Difference Learning(TD-Learning) is a kind of reinforcement learning and is able to adjust weights automatically.
  - The goal of TD-Learning is to predict the actual return R_t
  - The way it adjusts the weight is through: $V(s_t) = V(s_t) + \alpha\big(R_t - V(s_t)\big)$
  - **TD(0):** $\boldsymbol{V(s_t) = V(s_t) + \alpha(r_{t+1} + V(s_{t+1}) - V(s_t))}$
- According to [1], since 2048 is a stochastic game (random tile generation after player's move). Training by afterstate is a better choice. Check *Methodology* section for more details.
- Please hand in your source code and report, and demo to TAs.

Game Environment:

- 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.

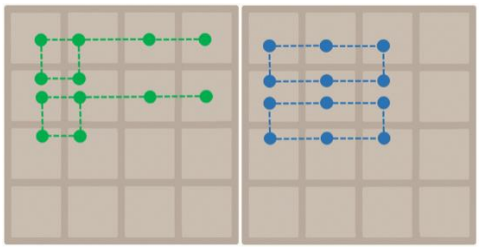
2048

- A sample of two-step state transition



Configuration:

- Learning rate: 0.1
  - Train the network 500k ~ 1M games
- N-Tuples Pattern: four 6-tuples
  -



Requirements:

1. Set patterns, learning rates, seed, etc.
2. Implement **learning::select_best_move**(const board&).
3. Implement **learning::update_episode**(std::vector<state>&, const float&).

Methodology:

You should check [1] for more details.

A pseudocode of a game engine (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'')
    $score \leftarrow score + r$
    $s \leftarrow s''$
  if LEARNING ENABLED **then**
    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)-afterstate

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

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

Scoring Criteria (Demo; Non-Demo):

- Strength (30%; 40%)
  - The 2048-tile reaching rate in 1000 games, $\lceil \text{winrate}_{2048} \rceil$
- Report (50%; 60%)
  - Introduction and experiment setup (10%)
  - Describe how you implement learning::select_best_move (30%)
  - Describe how you implement learning::update_episode (30%)
  - Statistic charts include following data (20%)
    - Winning rate and average score of standard configuration
    - Winning rate and average score of your configuration (if exists)
  - Discussion (10%)
- Demo (20%; 0%)
  - Run your program

Extra Bonus:

- Implement the original training method described in [1] (Figure 3), and compare the performance with the modified backward training method mentioned above.

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.