LAB 1

TA 張孝全

Deadline: 2024/09/22(Sun) 23:59

Demo: 2024/09/23(Mon)

In this lab,

Must use sample code,
otherwise no credit.

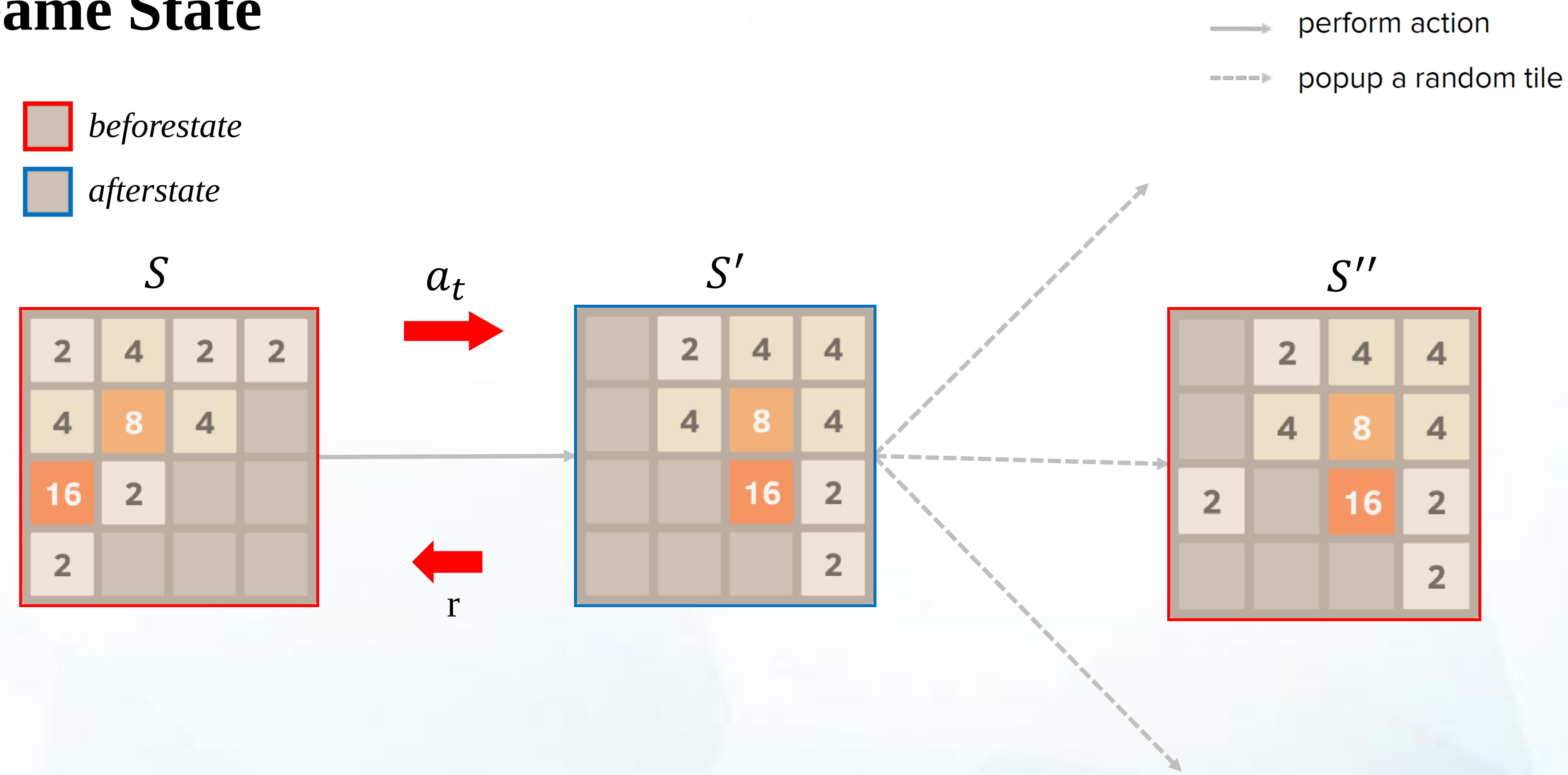
Outline

- 2048 Game Rule
- Game State
- Temporal Difference Learning
- n-tuple Network
- Sample Code
- Scoring Criteria
- Reminders

2048 Game Rules

popup: **2** (90%), **4** (10%)

Game State



perform action
popup a random tile
beforestate
afterstate
S
a_t
r
S'
S''
2 4 2 2
4 8 4
16 2
2
2 4 4
4 8 4
16 2
2
2 4 4
4 8 4
2 16 2
2

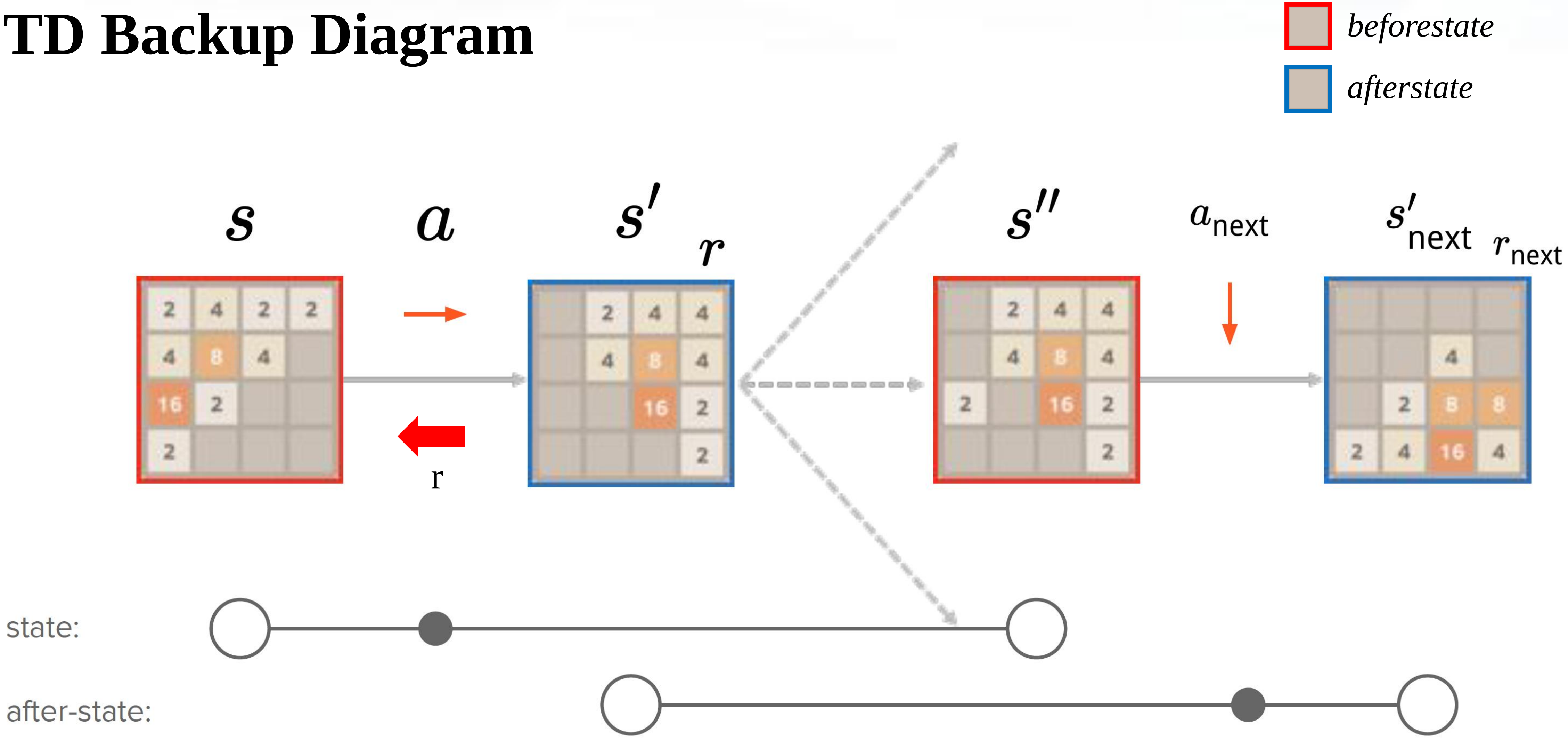
Temporal Difference Learning (TD)

For each episode,

```
Initialize (before-)state s
While s is not terminal do
  a ← argmax_a' EVALUATE(s, a')
  r, s', s'' ← MAKE_MOVE(s, a)
  STORE(s, a, r, s', s'')
  s ← s''
End While
For (s, a, r, s’, s’’) from terminal down to initial do
  LEARN_EVALUATION(s, a, r, s', s'')
End For
```

perform TD backup

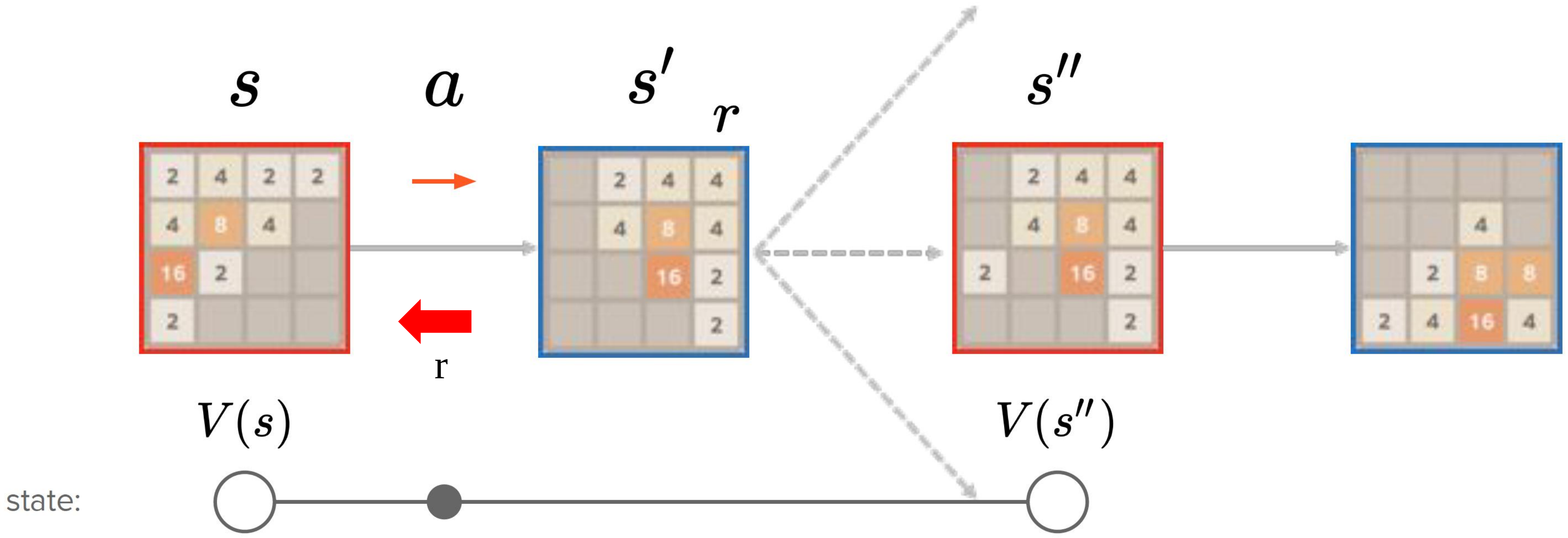
TD Backup Diagram



TD Backup: State

beforestate

afterstate

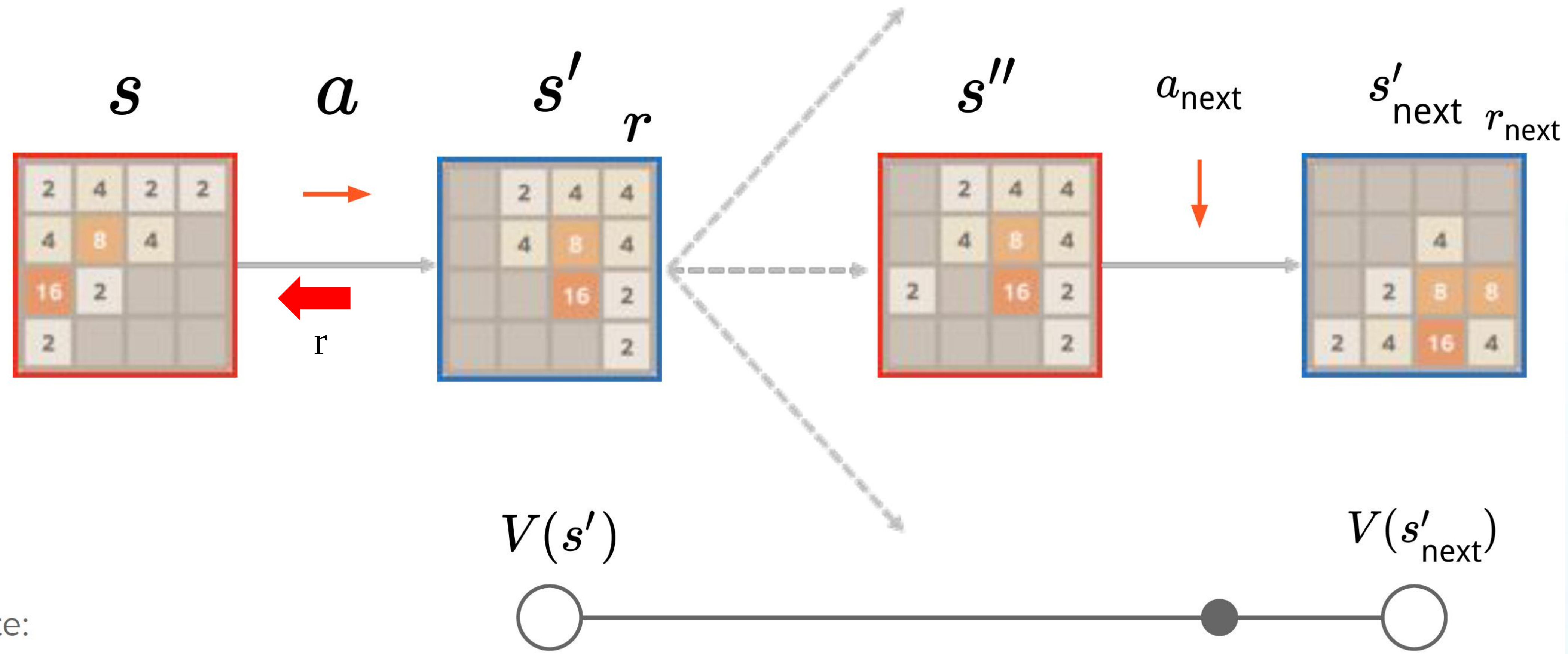


$$V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$$

TD Backup: After-State

beforestate

afterstate



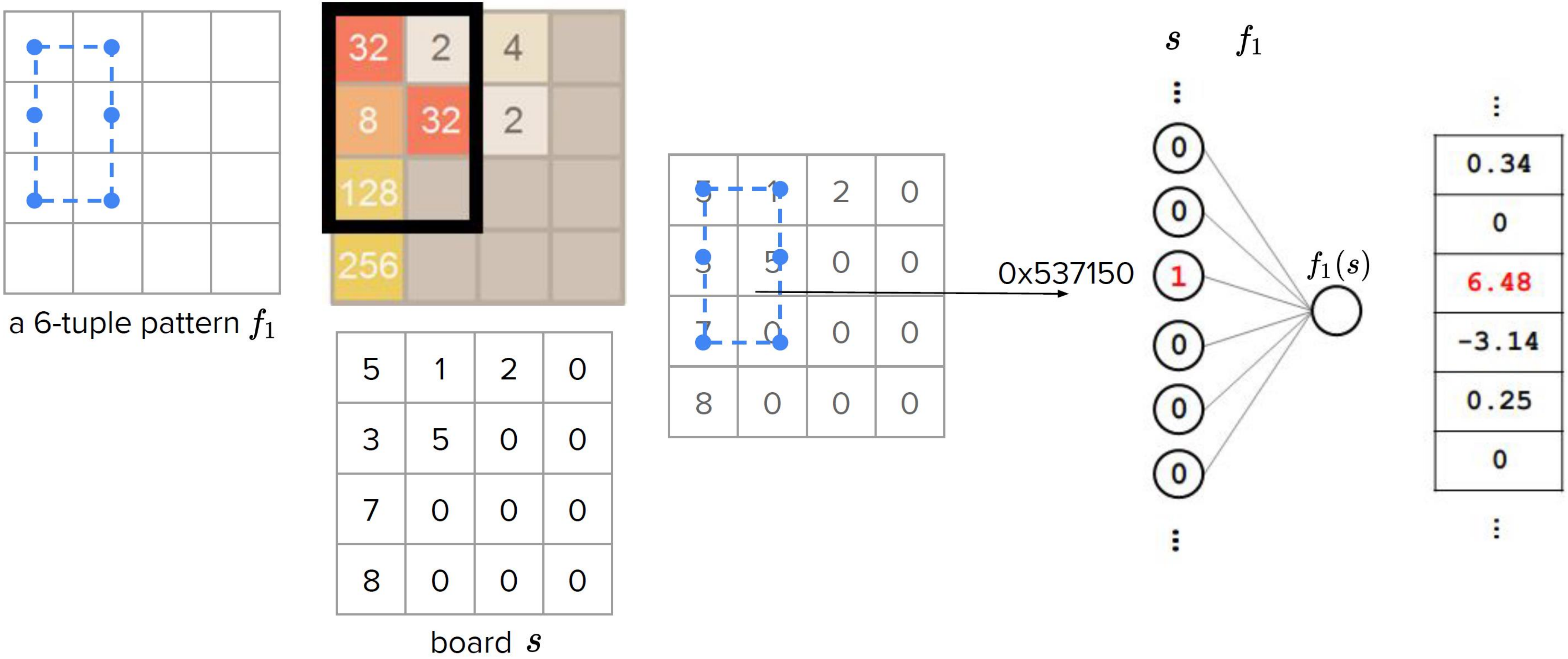
after-state:

$$V(s') \leftarrow V(s') + \alpha(r_{\text{next}} + V(s'_{\text{next}}) - V(s'))$$

Why use n-tuple network?

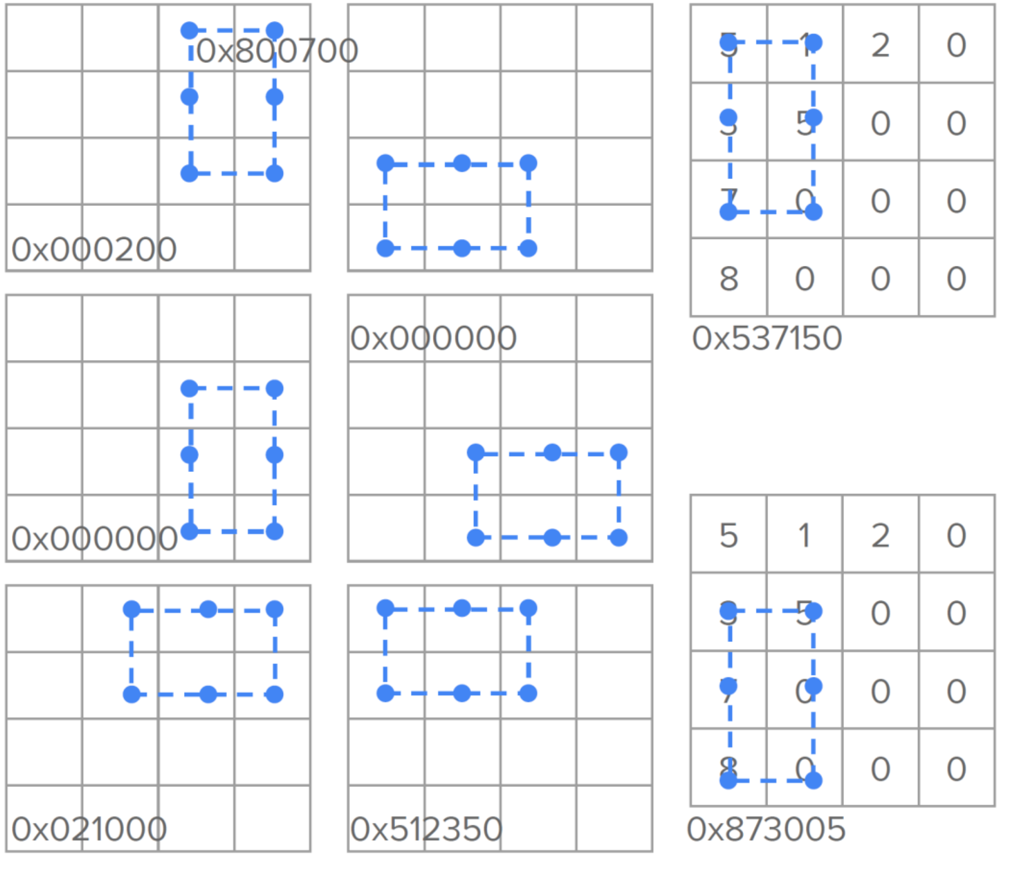
- The expected score/return G_t from a board S
- But, #states is huge
  - About 17^{16} ($=10^{20}$).
    - Empty ($\rightarrow$ 0), 2 ($=2^1 \rightarrow$ 1), 4 ($=2^2 \rightarrow$ 2), 8 ($=2^3 \rightarrow$ 3), …, 65536 ($=2^{16} \rightarrow$ 16).
- Need to use value function approximator.

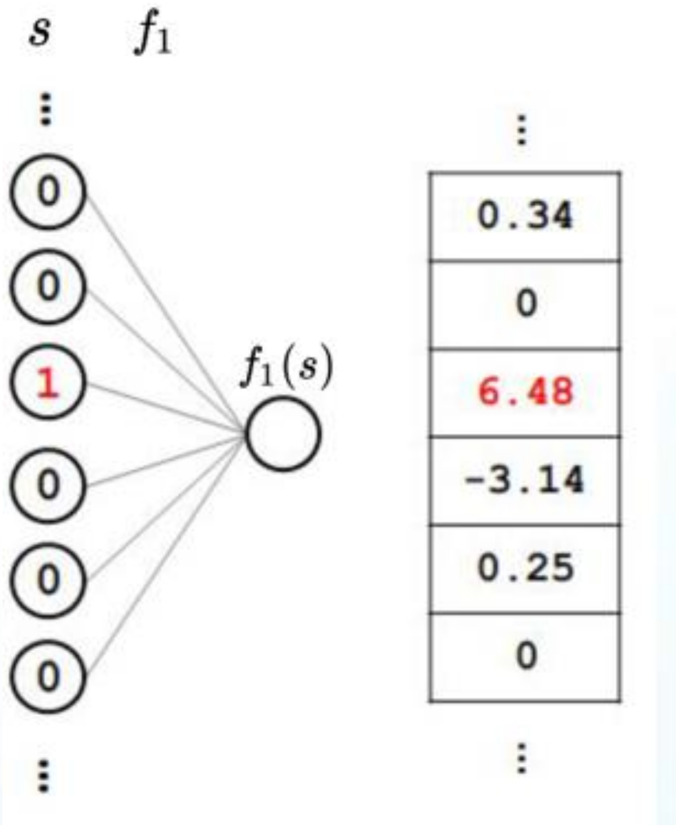
Example: 2048 with n-tuple network



All Isomorphism

- Rotations and Reflections
- The sum of the eight values can represents the board.

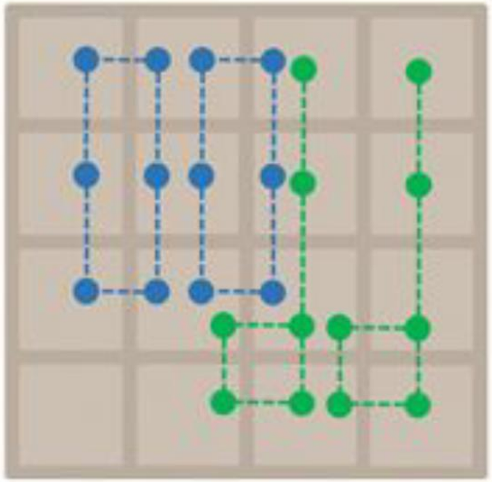


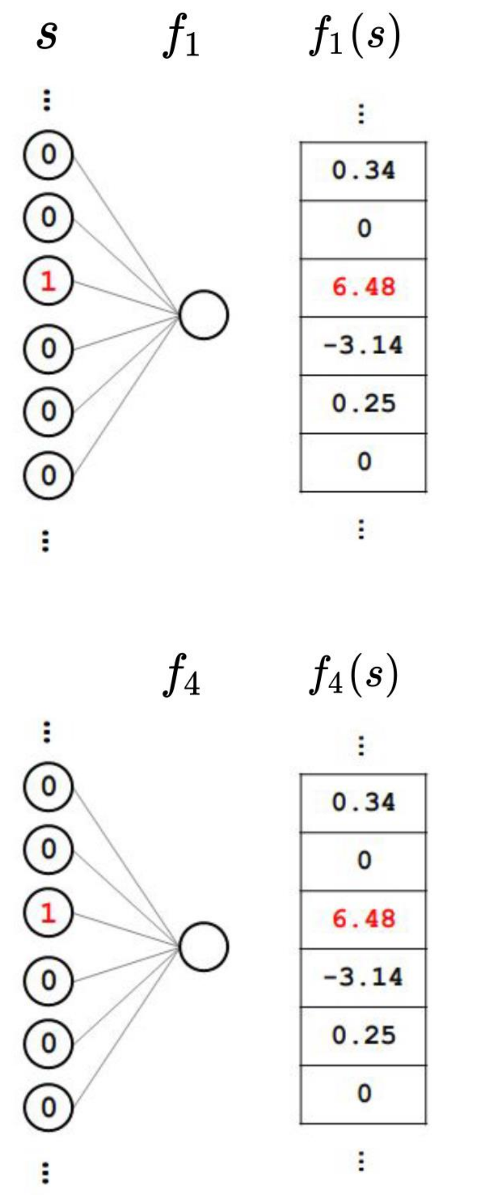


Multiple n-tuple

- Example: 4 kinds of 6-tuple.

$$V(s) = f_1(s) + f_2(s) + f_3(s) + f_4(s)$$





Sample Code

- Implement V(state)
    - Compile with C++11 support
    - g++ -std=c++11 -O3 -o 2048 2048.cpp

Training:

```
// restore the model from file
tdl.load("");

// train the model
std::vector<state> path;
path.reserve(20000);
for (size_t n = 1; n <= total; n++) {
  board b;
  int score = 0;

  // play an episode
  debug << "begin episode" << std::endl;
  b.init();
  while (true) {
    debug << "state" << std::endl << b;
    state best = tdl.select_best_move(b);
    path.push_back(best);

    if (best.is_valid()) {
      debug << "best " << best;
      score += best.reward();
      b = best.after_state();
      b.popup();
    } else {
      break;
    }
  }
  debug << "end episode" << std::endl;

  // update by TD(0)
  tdl.update_episode(path, alpha);
  tdl.make_statistic(n, b, score);
  path.clear();
}

// store the model into file
tdl.save("weights.bin");

return 0;
```

Save your model weight

Evaluating (demo):

Set total count to 1000 games

Load your model weight

```
int main(int argc, const char* argv[]) {
  info << "TDL2048-Demo" << std::endl;
  learning tdl;

  // set the learning parameters
  float alpha = 0.1;
  size_t total = 1000;
  unsigned seed;
  __asm__ __volatile__ ("rdtsc" : "=a" (seed));
  info << "alpha = " << alpha << std::endl;
  info << "total = " << total << std::endl;
  info << "seed = " << seed << std::endl;
  std::srand(seed);

  // initialize the features
  tdl.add_feature(new pattern({ 0, 1, 2, 3, 4, 5 }));
  tdl.add_feature(new pattern({ 4, 5, 6, 7, 8, 9 }));
  tdl.add_feature(new pattern({ 0, 1, 2, 4, 5, 6 }));
  tdl.add_feature(new pattern({ 4, 5, 6, 8, 9, 10 }));

  // restore the model from file
  tdl.load("weights.bin");

  // train the model
  std::vector<state> path;
  path.reserve(20000);
  for (size_t n = 1; n <= total; n++) {
    board b;
    int score = 0;
```

Scoring Criteria

Show your work, otherwise no credit will be granted.

- Report (20% + Bonus 20%)
- Performance (80%)
    - The 2048-tile win rate in 1000 games,[winrate_{2048}].(60%)
    - Questions. (20%)

```
1000    mean = 21355.2  max = 64492
        128     100%    (0.1%)
        256     99.9%   (1.4%)
        512     98.5%   (11.6%)
        1024    86.9%   (51.2%)
        2048    35.7%   (34.6%)
        4096    1.1%    (1.1%)
```

Reminders

- You can design your n-tuple.
- You should avoid using CNN in this lab.
- You have to load your weight while demo.

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