Reinforcement Learning

Concepts and Web Applications

KACZMAREK ADRIEN 05/19/2020

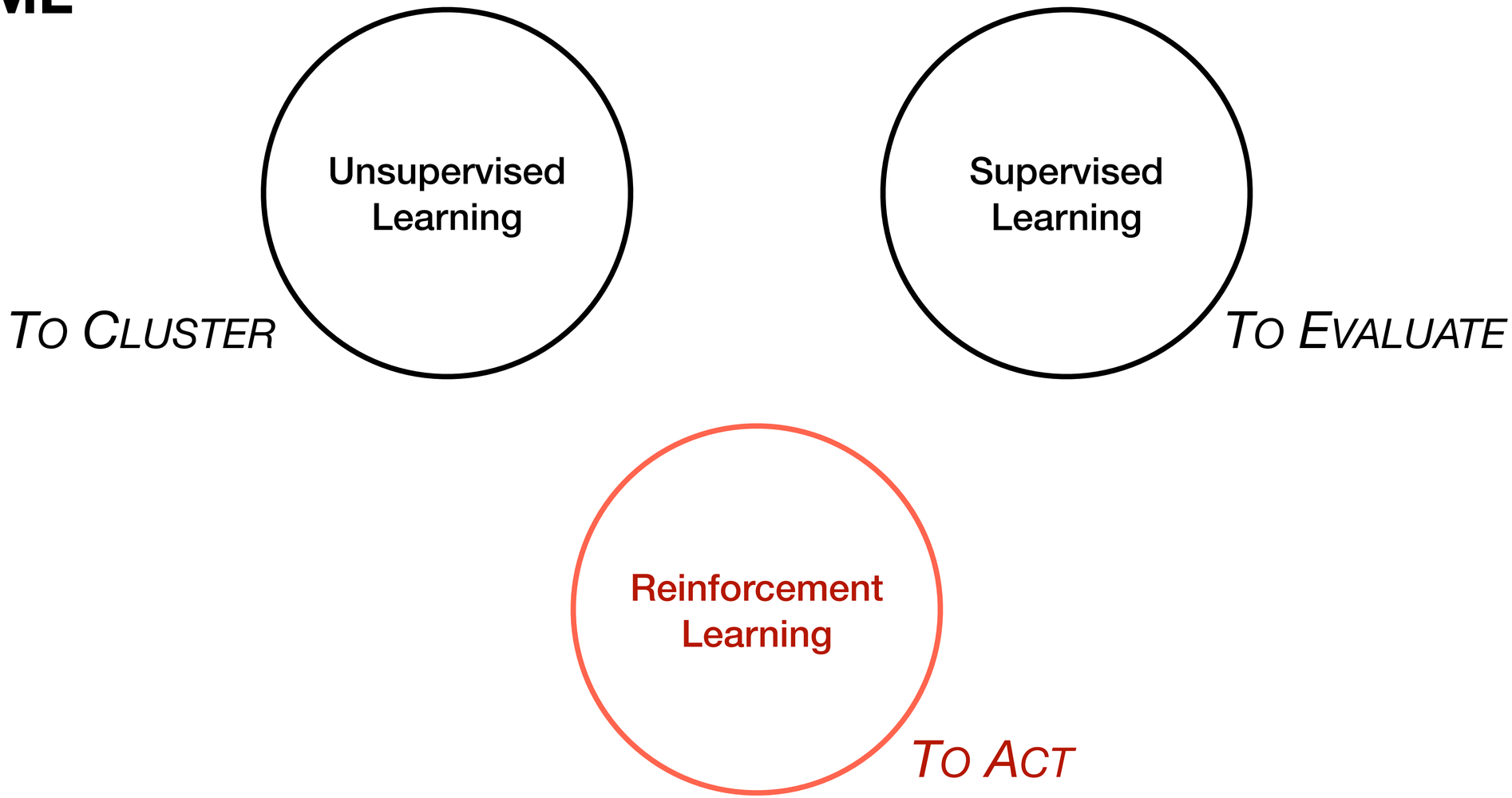
Overview

Brief look into Reinforcement Learning (RL)

1. What is RL?
2. How is RL applied to the Web?
3. Code Project
4. Why did I choose this topic?

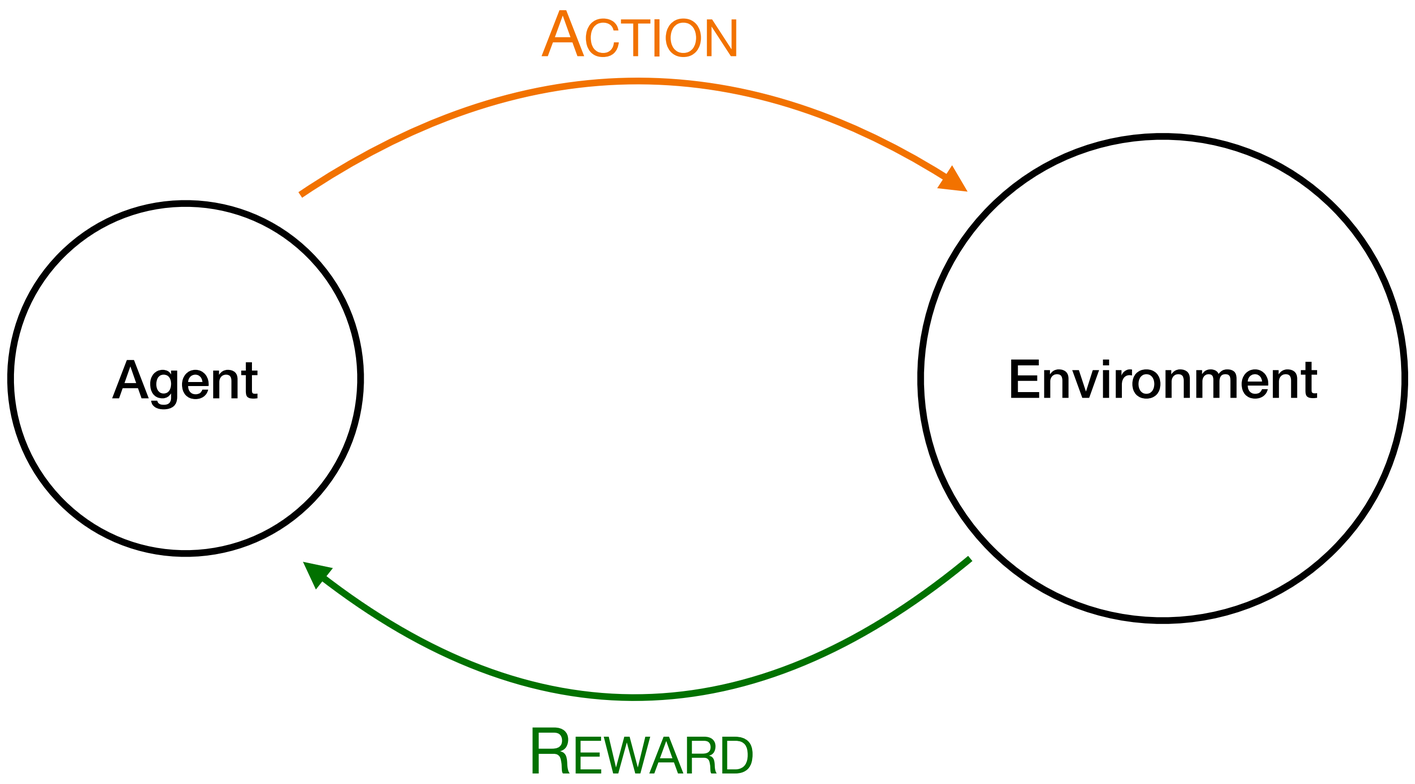
What is RL?

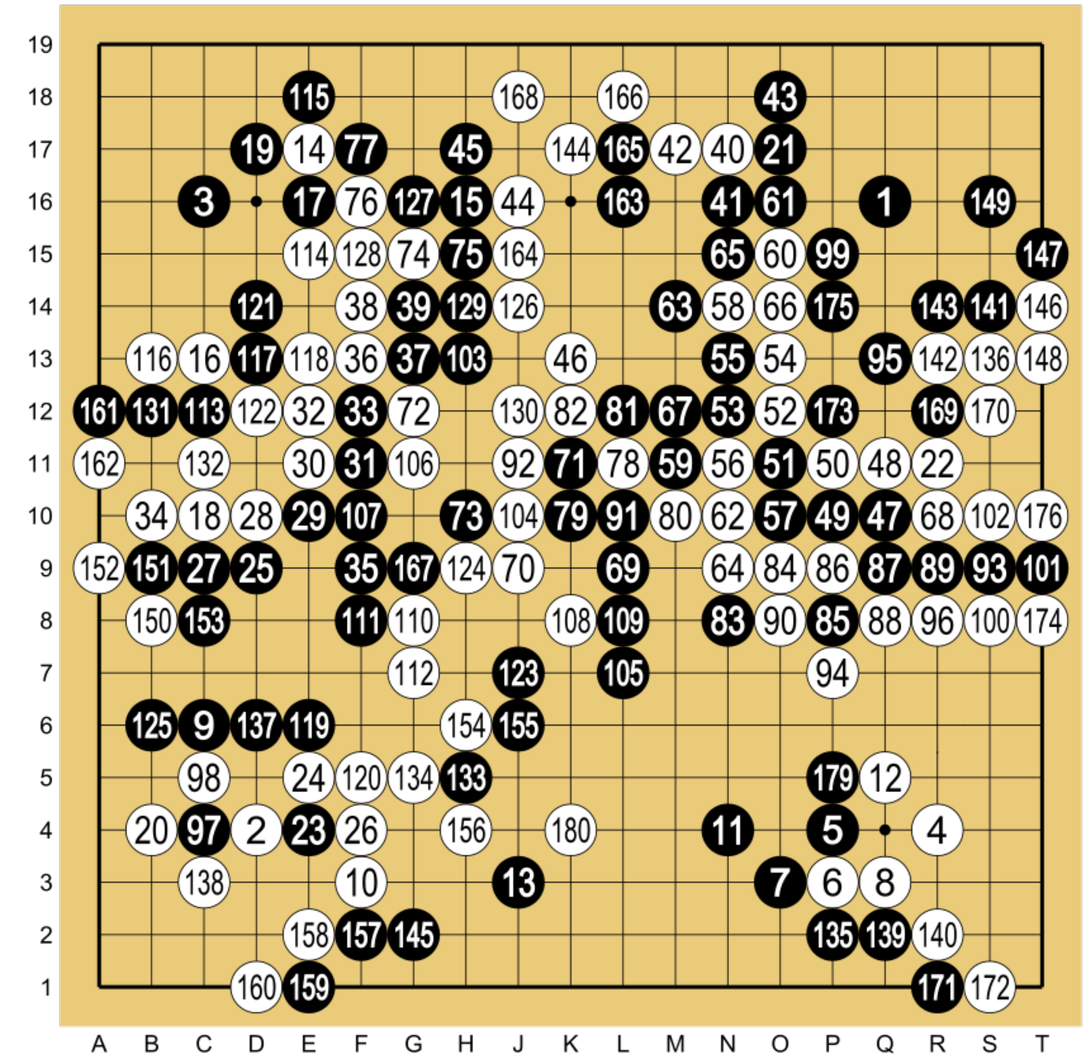
RL & ML



What is RL?

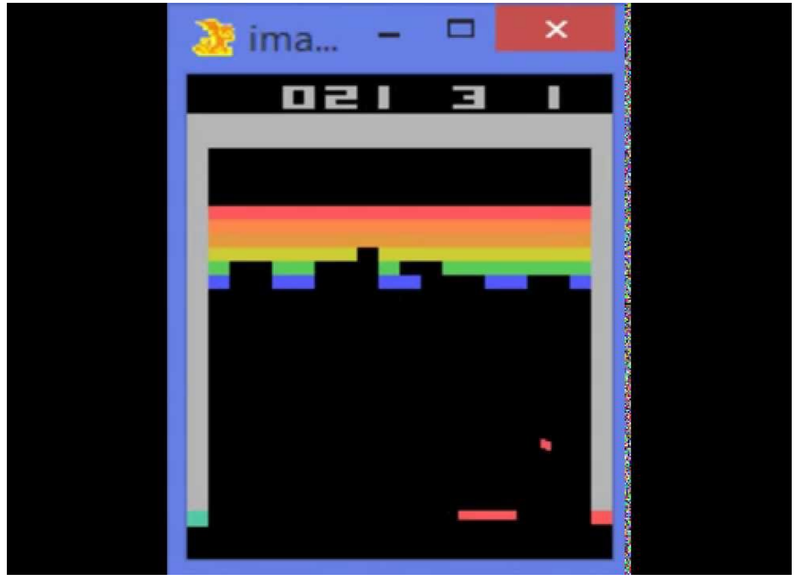
Agent & Environment





Lee Sedol (W) vs AlphaGo (B) - Game 4

177 at 51 178 at 57



How is RL applied to the Web?

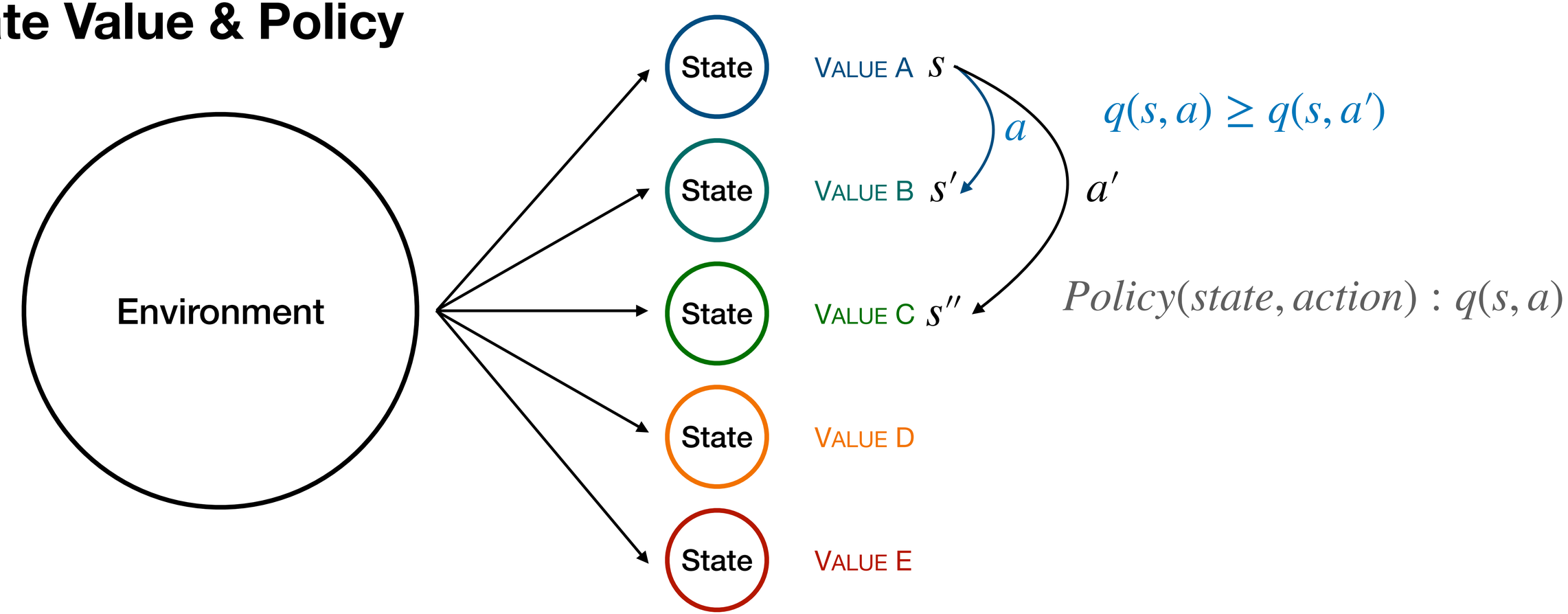
Trial and error

- Web Spiders (~2000)
- Web Recommendation (~2016)
- Internet Cogestion Control (~2019)

Code Project - Frozen Lake

Code Project

State Value & Policy

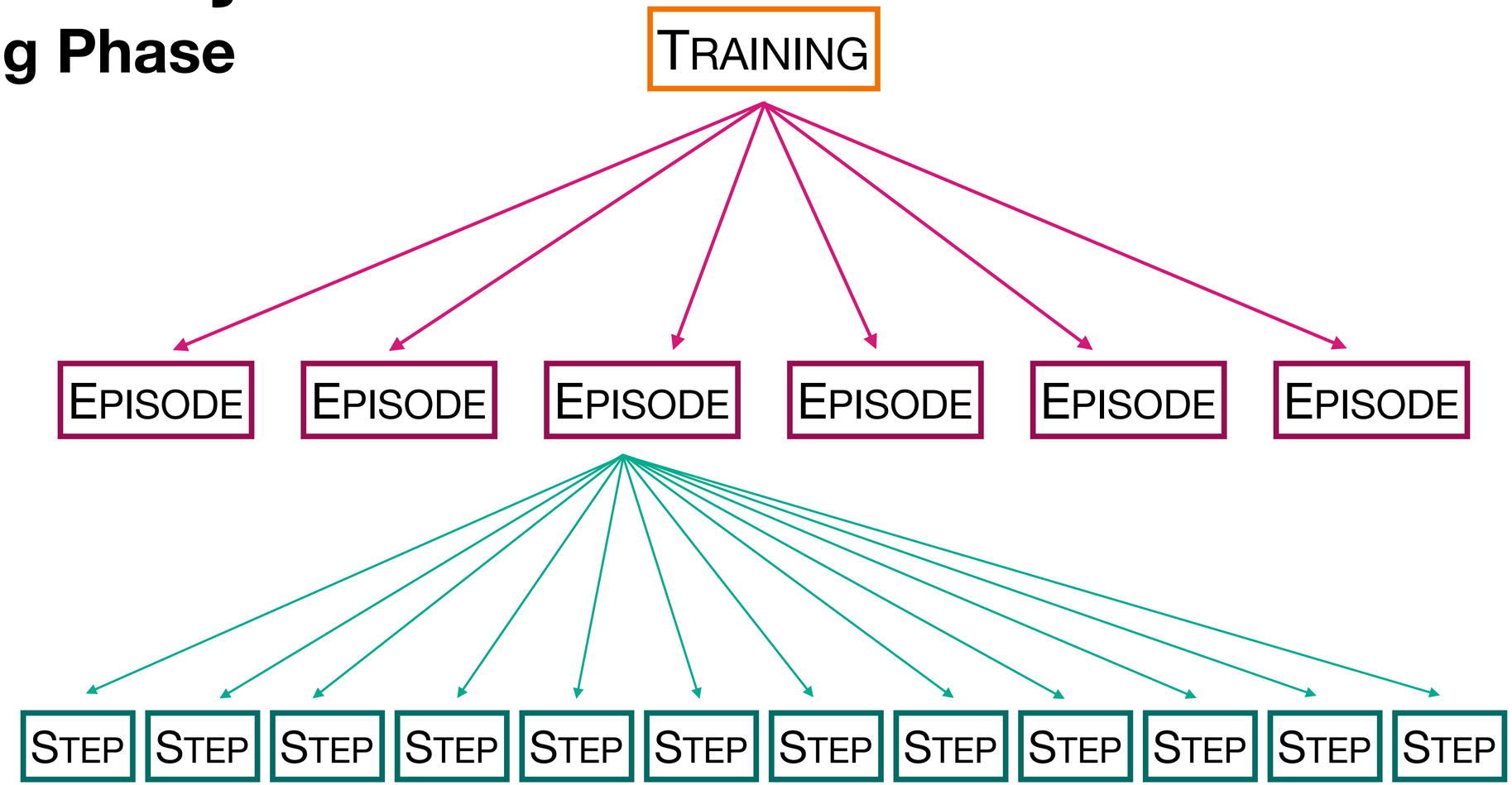


$$q_*(s,a) = E\left[(1-\alpha)q_*(s,a)+\alpha\left(R_{t+1}+\gamma \max_{a'} q_*(s',a')\right)\right]$$

Bellmann's Equation

Code Project

Training Phase



Logic

```
// Helper

function modulo(n, m){
    return ((n % m) + m) % m;
}

// Environment
function Environment() {…
};

Environment.prototype.generate_random_map = function(p_max) {…
};

Environment.prototype.reset = function(use_random, make=true) {…
};


Environment.prototype.step = function(action) {…
};

Environment.prototype.move = function(direction) {…
};

Environment.prototype.render = function(type, object) {…
};
```

```
Environment.prototype.step = function(action) {
    let new_state = 0;
    let reward = 0;
    let done = false;
    let info = "";

    const action_index = this.action_space.indexOf(action);

    if (Math.random() > 1 - this.slip_p) { // Slip
        if (Math.random() > .5) {
            new_state = this.move(this.action_space[modulo(action_index - 1, 4)])
        } else {
            new_state = this.move(this.action_space[modulo(action_index + 1, 4)])
        };
    } else { // Don't slip
        new_state = this.move(action);
    };

    reward = this.reward_mapping[this.map[new_state]][0];

    if (this.reward_mapping[this.map[new_state]][1]) {
        done = true;
    }

    return {'new_state':new_state, 'reward':reward, 'done':done, 'info':info};
};
```

```
// Environment create in main.js

// helper modulo already create

// Agent
function Agent() {...
}

Agent.prototype.train = function(text_renderer, use_random_generator,
    use_clever_state) {...
};

Agent.prototype.exploit = function(use_clever_state) {...
};

Agent.prototype.get_clever_state = function() {...
};

Agent.prototype.get_average_rewards = function(rewards_all_episodes, count) {...
};

Agent.prototype.get_render_average_rewards = function(rewards_all_episodes,
    count) {...
};

Agent.prototype.get_render_q_table = function() {...
};
```

```
Agent.prototype.train = function(text_renderer, use_random_generator, use_clever_state) {
    // Initialize Q-Tables
    this.state_space_size = env.height * env.width;
    this.q_table = [];
    if (use_clever_state) { ...
    }
    else {
        for (let i = 0; i < this.state_space_size; i++) {
            this.q_table.push([]);
            for (let j = 0; j < this.action_space_size; j++) {
                this.q_table[i].push(0);
            };
        };
    };

    // Adjust the number of episode
    let last_remake = 0;
    if (use_clever_state) {
        this.num_episodes = this.num_episodes_object.clever;
    }
    else {
        this.num_episodes = this.num_episodes_object.basic;
    };


    // Q-learning Algorithm

    const rewards_all_episodes = [];

    let exploration_rate = this.max_exploration_rate;

    for (let episode = 0; episode < this.num_episodes; episode++) {
        let state = env.reset(use_random_generator, false);

        // If use_clever_state change the encoding of the environment
        if (use_clever_state) { ...
        };

        let done = false;
        let rewards_current_episode = .0;

        let step = 0;
        while (step++ < this.max_steps_per_episode && !done) {
            let action_index;
            // Exploration vs. Exploitation
            if (Math.random() > exploration_rate) { // Exploitation
                action_index = this.exploit(use_clever_state);
            }
            else { // Exploration, Move at Random
                action_index = Math.floor(Math.random() * this.action_space_size);
            };
            action = env.action_space[action_index];

            let step_return = env.step(action);

            let new_state = step_return.new_state;
            if (use_clever_state) {
                new_state = this.get_clever_state();
            };

            // Update Q-table Q(s,a)
            this.q_table[state][action_index] = this.q_table[state][action_index] * (1 - this.learning_rate) + this.learning_rate * (step_return.reward + this.discount_rate * Math.max(...this.q_table[new_state]));

            state = new_state;
            done = step_return.done;

            // Only keep reward due to success (human reading facilitator)
            if (done && step_return.reward > 0) {
                rewards_current_episode += step_return.reward;
            };
        };


        // Exploration rate decay
        exploration_rate = this.min_exploration_rate + (this.max_exploration_rate - this.min_exploration_rate) * Math.exp(- this.exploration_decay_rate * episode);
        rewards_all_episodes.push(rewards_current_episode);
    };

    return rewards_all_episodes;
};
```

```
// Update Q-table Q(s,a)
this.q_table[state][action_index] = this.q_table[state][action_index] *
    (1 - this.learning_rate) + this.learning_rate * (step_return.reward
    + this.discount_rate * Math.max(...this.q_table[new_state]));
```

$$q_*(s,a) = E\left[(1-\alpha)q_*(s,a)+\alpha\left(R_{t+1}+\gamma \max_{a'} q_*(s',a')\right)\right]$$

```
Agent.prototype.exploit = function(use_clever_state) {
    let action_index = 0;
    let state = 0;
    if (use_clever_state) {
        state = this.get_clever_state();
        action_index = this.q_table[state].indexOf(Math.max(...this.q_table[state]))
            ;
    }
    else {
        state = env.state;
        action_index = this.q_table[state].indexOf(Math.max(...this.q_table[state]))
            ;
    };

    return action_index;
};

Agent.prototype.get_clever_state = function() {…
};

Agent.prototype.get_average_rewards = function(rewards_all_episodes, count) {…
};

Agent.prototype.get_render_average_rewards = function(rewards_all_episodes, count) {
…
};
```

(Hyper) Parameters

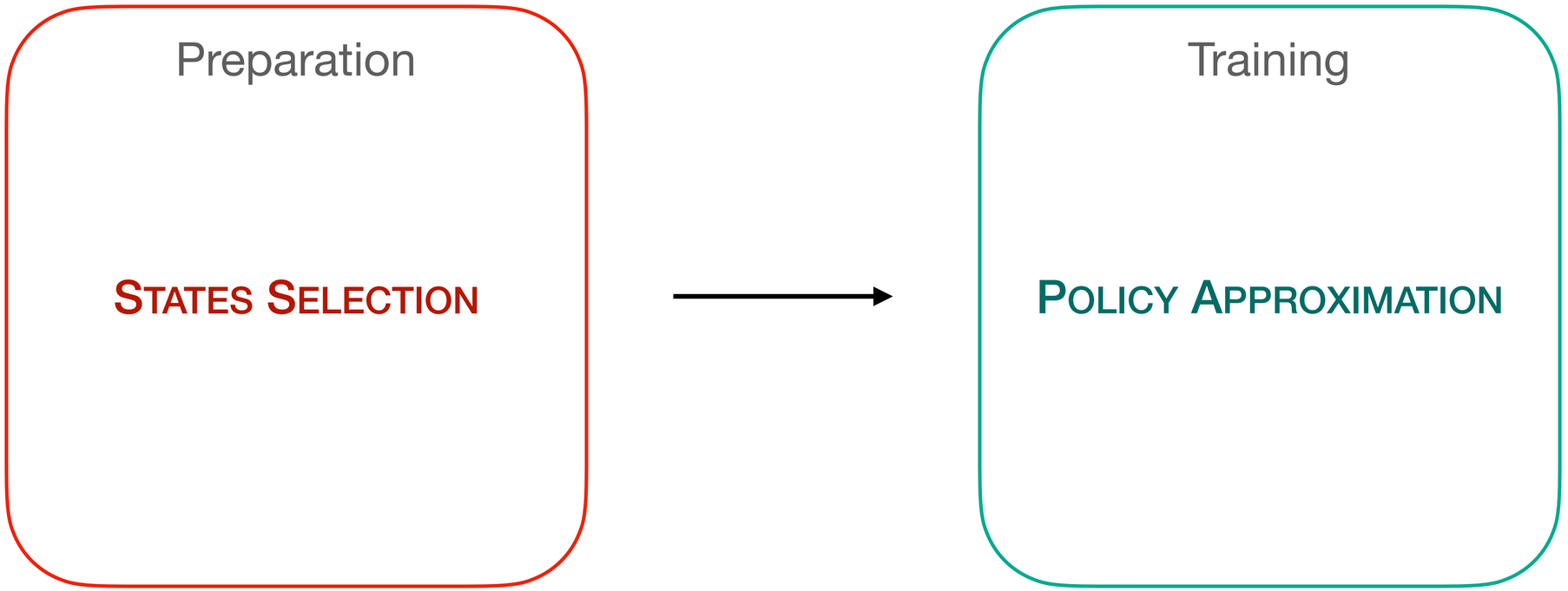
```
// Agent
function Agent() {
    // Parameters
    this.num_episodes_object = {basic: 1e4, clever: 1e5};
    this.max_steps_per_episode = 100;

    this.learning_rate = 0.1;
    this.discount_rate = 0.99;

    this.max_exploration_rate = 1;
    this.min_exploration_rate = 0.01;
    this.exploration_decay_rate = 0.001;

    // Random
    this.random_remake = 1e3;

    // Q-table
    this.action_space_size = env.action_space.length;
    this.state_space_size = 0;

    this.num_episodes = this.num_episodes_object.basic;
    this.q_table = [];
}
```

```
// Environment
function Environment() {
    this.name = "Frozen Lake";

    this.action_space = ['up', 'right', 'bottom', 'left'];

    this.slip_p = .66;

    this.width = 4;
    this.height = 4;

    this.base_map = "   SFFF\
                        FHFH\
                        FFFH\
                        HFFG";


    this.map = '';

    this.reward_mapping = { 'S' : [0.001, false],
                            'F' : [0.001, false],
                            'H' : [-.5, true] ,
                            'G' : [1, true]};

    this.state = 0;

    this.reset();
};
```

Action … !

Code Project

Summary



Discrete

Continuous

Complete

...

Q-Table

Deep Neural Network

ACS PPO

...

Why did I choose this topic?