CAPSTONE DESIGN II

강화학습 기반의 자율주행 전기차 이동충전소

컴퓨터공학과 2018110646 김연수

Contents

Introduction

1. Background

○ Problem

- 국내에 보급된 전기차에 비해 전기차 충전소 인프라가 부족
- 지역별 전기차 충전소 분포가 고르지 않음

2. Importance

○ 강화학습을 활용하여 실시간으로 이동식 전기차 충전소가 최적의 경로로 자율주행 하여 전기차를 충전

○ 이를 통해 전기차 충전소 인프라 부족 현상을 유연하게 해결하고자 함

Related Works

1. MADDPG

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

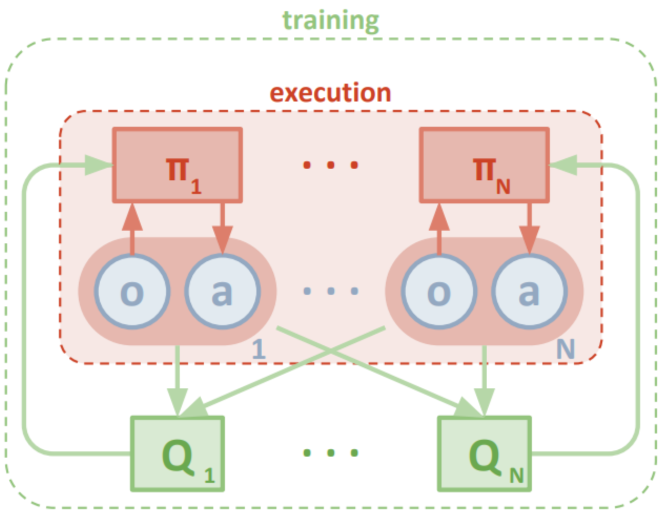
for episode = 1 to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma \, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a_1', \ldots, a_N')|_{a_k' = \boldsymbol{\mu}_k'(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S} \sum_j \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta_i' \leftarrow \tau \theta_i + (1 - \tau)\theta_i'$$

  end for
end for



Related Works

2. Auction Theory

/* Bidding Submission of d_i */

Each buyer d_i requests the energy trading and provides its bid vector, $\mathbb{B}_i$

/* Winning Bid Determination at s_j */

$x_{ij}^{temp} = 0, p_j^t = 0, \forall d_i \in \mathcal{D}; s_j \in \mathcal{S}; W_j^{can} = \emptyset.$

Determine the set of feasible buyers $W_j^{temp} = \{d_i | e_i \leq E_j, t_i \leq T_j, \forall d_i \in \mathcal{D}\}$.

Sort the bids of buyers $d_i \in W_j^{temp}$ in non-increasing order: $W_j^{order} = (d_{j1}, d_{j2}, ..., d_{jK})$ such that $b_{j_1 j} \geq b_{j_2 j} \geq ... \geq b_{j_K j}$ with $K = |W_j^{temp}|$.

Pick out $|c_j|$ bidders having the highest bids

$$W_j^{cons} = \begin{cases} (d_{j1}, d_{j2}, ..., d_{jc_j}), & \text{if } c_j < K, \\ W_j^{order}, & \text{otherwise.} \end{cases}$$

if $\sum_{d_i \in W_j^{cons}} e_i \leqslant E_j$ **then**

$W_j^{can} = W_j^{cons}$; $x_{ij}^{temp} = 1, \forall d_i \in W_j^{can}$.

$p_j^t = b_{mj}$ where

$$m = \begin{cases} \arg\max_i \{d_i | d_i \in W_j^{order} \setminus W_j^{cons}\} & \text{if } c_j < K, \\ \arg\min_i \{d_i | d_i \in W_j^{order}\} & \text{otherwise.} \end{cases}$$

if $\sum_{d_i \in W_j^{cons}} e_i > E_j$ **then**

$h = \arg\max_h \sum_{h'=1}^{h} e_{h'} \leq E_j, \forall d_{h'} \in W_j^{cons}$.

$W_j^{can} = \{d'_h | 1 \leq h' \leq h\}; x_{h'j}^{temp} = 1.$

$p_j^t = b_{(h+1)j}$.

/* Final seller determination at d_i */

for $j = 1$ *to* S **do**

$x_{ij} = 0; p_i^d = 0.$

if $\sum_{j=1}^{M} x_{ij}^{temp} = 0$ **then**

$x_{ij} = 0, \forall j; p_i^d = 0.$

if $\sum_{j=1}^{M} x_{ij}^{temp} = 1$ **then**

for $j = 1$ *to* M **do**

if $x_{ij}^{temp} = 1$ **then**

$x_{ij} = 1; p_i^d = p_j^t.$

if $\sum_{j=1}^{M} x_{ij}^{temp} > 1$ **then**

for $j = 1$ *to* M **do do**

if $x_{ij}^{temp} = 1$ **then**

$U_{ij}^d = (v_{ij} - p_j^t) e_i.$

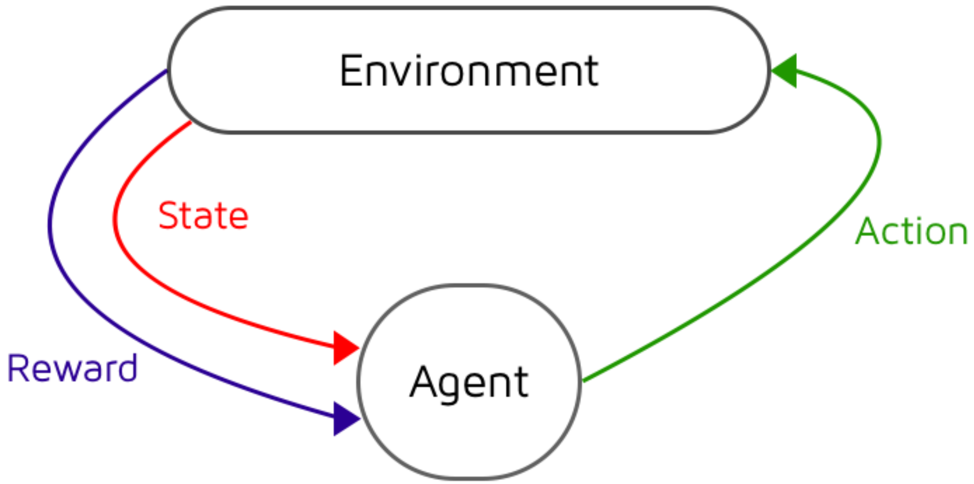
$j^* = \arg\max_j \{U_{ij}^d | x_{ij}^{temp} = 1, \forall j \in M\}.$

$x_{ij^*} = 1; p_i^d = p_{j^*}^t.$

Distributed Auction-Based Incentive Mechanism for Energy Trading between Electric Vehicles and Mobile Charging Stations

Method

1. Definition

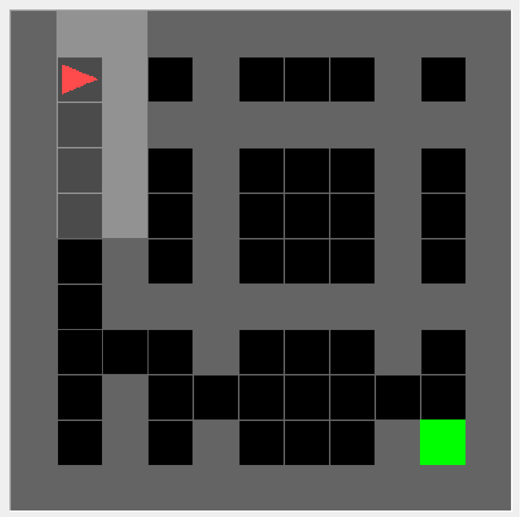


- Action Space
  - 주어진 환경에서 가능한 모든 action의 set
  - Move = {right, left, up, down, no-operation}
- Observation Space (State)
  - Environment의 현재 상태에 대한 정보
  - Agent인 전기차 충전소는 자신의 근처(상하좌우)에 있는 전기차를 알 수 있음
- Policy
  - Agent가 어떤 Action을 취할지 선택하는 Rule
  - 확률적(Stochastic) 접근 : Move_Probability = {0.175, 0.175, 0.175, 0.175, 0.3}

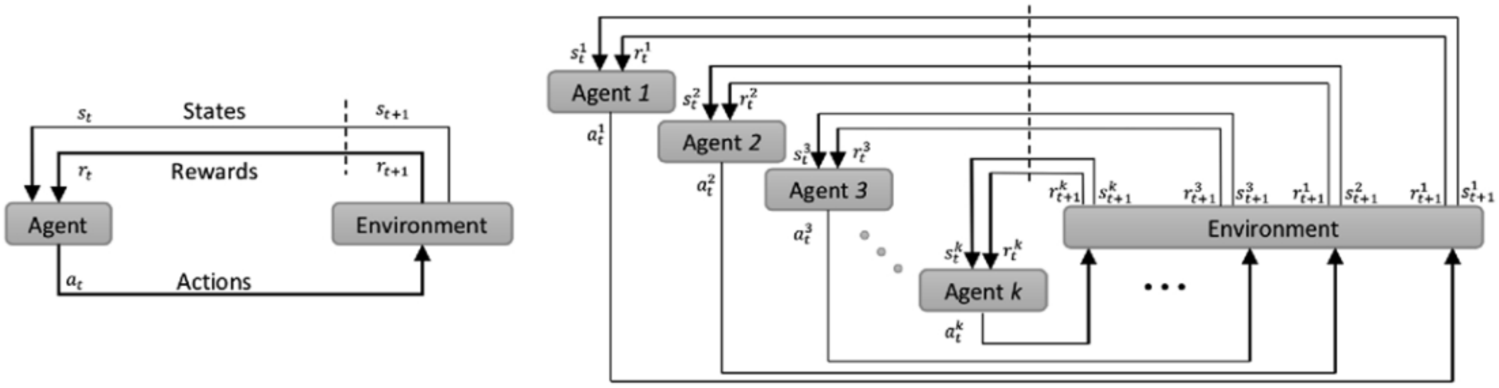
Method

2. Environment

- OpenAi/gym : gridworld



- Multi Agent



(a) Single Agent (b) Multi-Agents

Experiments

1. Scenario

1) 전기차 충전소가 action space를 기반으로 상태를 update 한다.

```
for agent_i, action in enumerate(agents_action):
    if not (self._agent_dones[agent_i]):
        self.__update_agent_pos(agent_i, action)
```

2) 전기차는 근처의 충전소가 어디에 위치해 있는지 정보를 바탕으로 가까운 충전소로 이동한다.

```
prey_move = None
if self._prey_alive[prey_i]:
    for _ in range(5):
        _move = self.np_random.choice(len(self._prey_move_probs), 1, p=self._prey_move_probs)[0]
        if self._neighbour_agents(self.__next_pos(self.prey_pos[prey_i], _move))[0] != 0: # PREY MUST GO IN NEIGHBOURHOOD OF PREDATOR
            neighbor_predator_idx = self._neighbour_agents(self.__next_pos(self.prey_pos[prey_i], _move))[1]
            if 0 < self.agent_avail_charging[neighbor_predator_idx[0]] - self.prey_need_charging[prey_i]:
                self.agent_avail_charging[neighbor_predator_idx[0]] -= self.prey_need_charging[prey_i] # charging : predator - prey
            prey_move = _move
            # print("res : ", self.agent_avail_charging[neighbor_predator_idx[0]])
            break
    prey_move = 4 if prey_move is None else prey_move  # default is no-op(4)
self.__update_prey_pos(prey_i, prey_move)
```

Experiments

1. Scenario

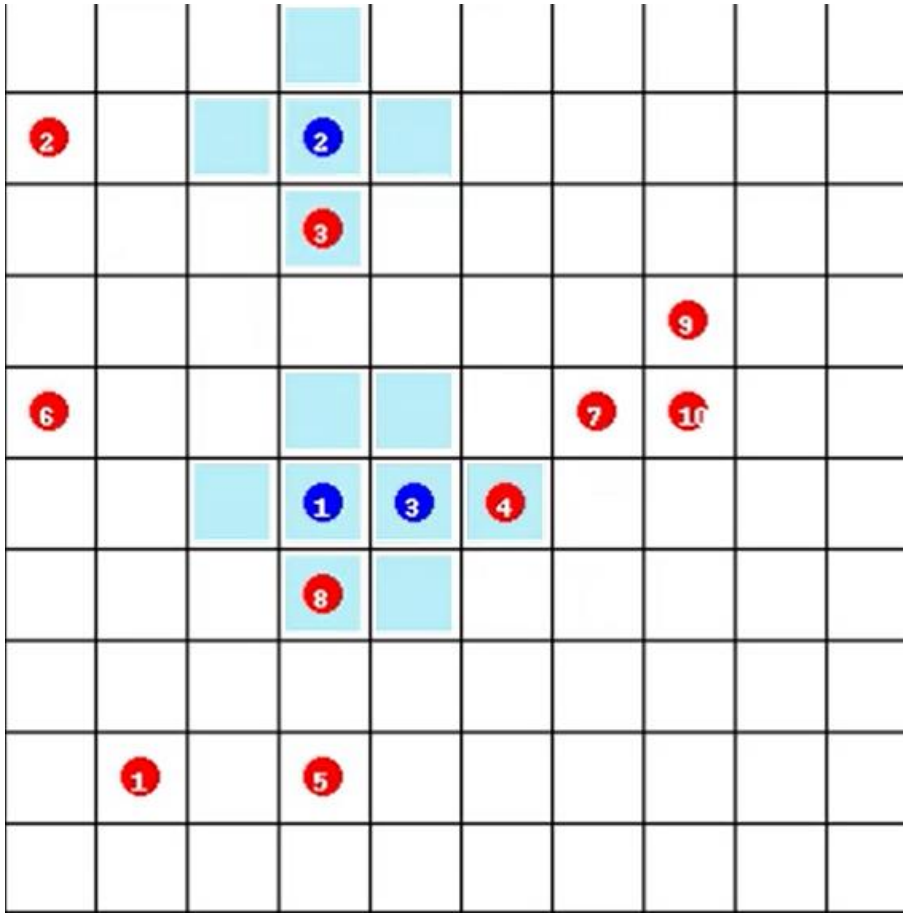
3) 만약 충전 대상에 대한 경쟁상황이 발생한다면, auction theory를 적용한다.

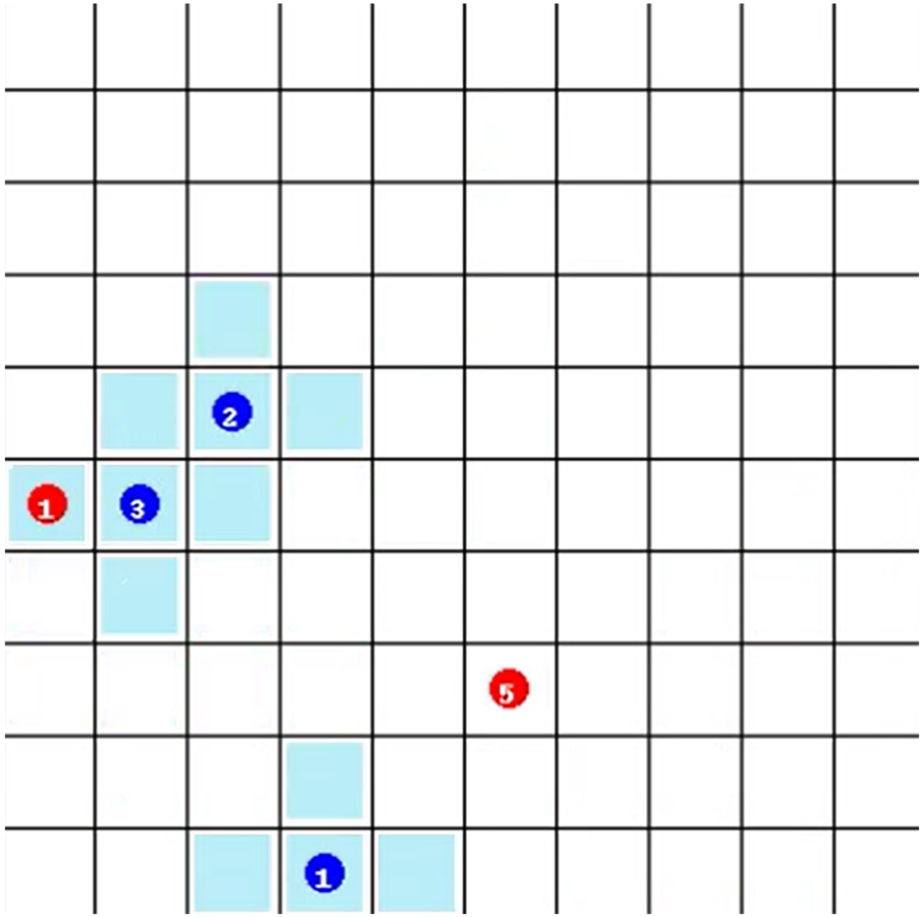
```
# auction theory
if (predator_neighbour_count > 1):
    winner_agent_idx = 0
    if (self.price[n_i[0]] < self.price[n_i[1]]):
        winner_agent_idx = 1
    if 0 < self.agent_avail_charging[winner_agent_idx] - self.prey_need_charging[prey_i]:
        self.agent_avail_charging[winner_agent_idx] -= self.prey_need_charging[prey_i]
```

4) Step을 진행하면서 1,2,3 번 과정을 반복한다.

Experiments

2. Output





Conclusion

1. 기대효과

- 전기차 충전소 인프라의 시간, 공간적 제약 극복
- 전기차 충전소가 최적의 경로로 자율주행

2. 추후 연구 계획

- 모델 경량화
- 임베디드 환경에서 실험