Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

백승언

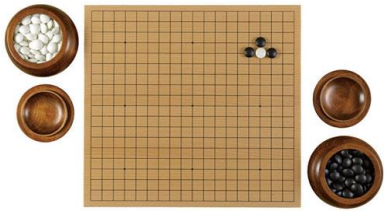
19 April, 2021

Contents

Introduction

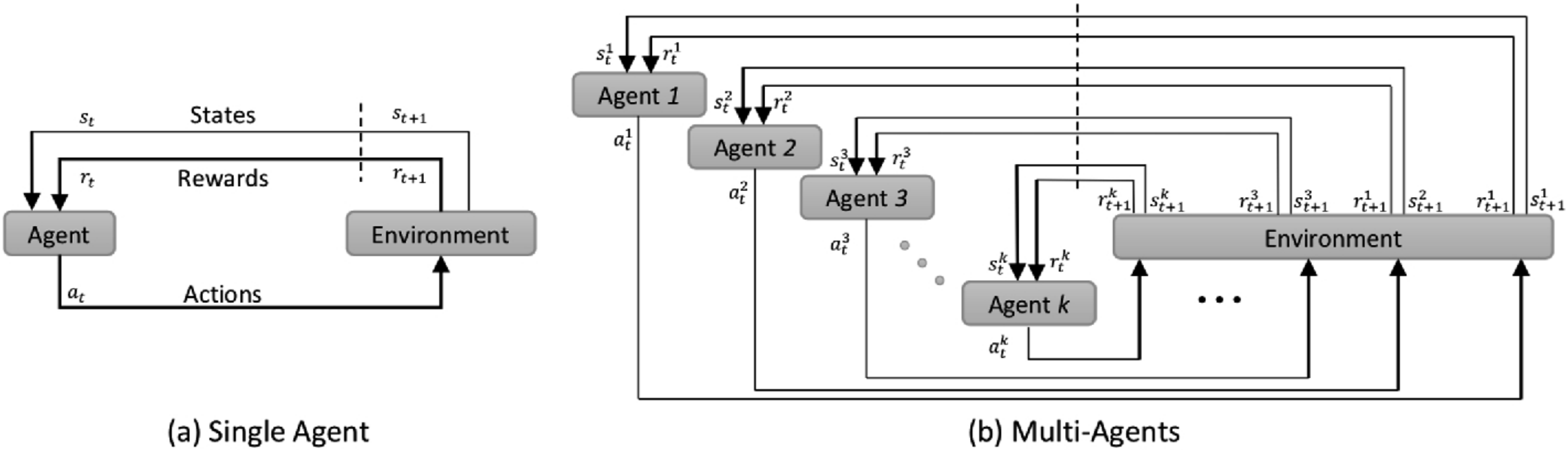
Advances in Reinforcement Learning in sequential decision-making problems

- **Milestones in games**
  - Backgammon (1992) ➔ TD(λ) algorithm
  - Chess (1997) ➔ TD(λ) algorithm
  - Atari game (2013) ➔ DQN
  - Go (2016) ➔ MC Tree search
  - DOTA2 (2019) ➔ PPO



Multi-Agent Reinforcement Learning(MARL) (I)

- **Sequential decision-making problem of multiple agents that operate in a shared/common env**
  - MARL aims to optimize each agent's own long-term return by interacting with the **env and other agents**
  - MARL in multi-agent systems has potential to extend in many subareas
    - Autonomous vehicle, finance, sensor/communication networks and social science

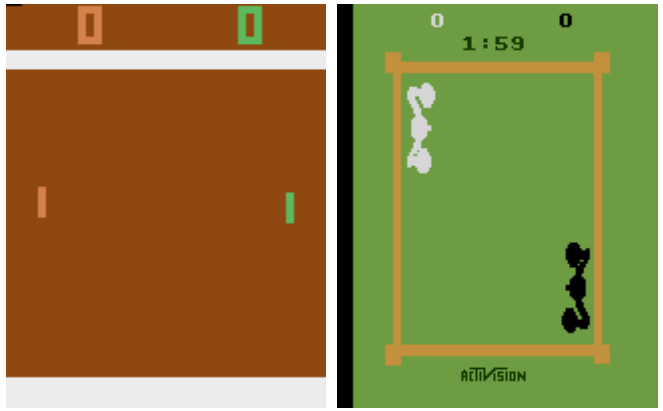


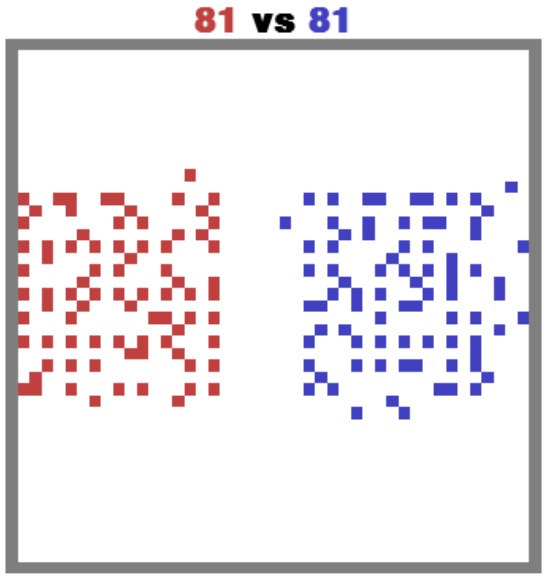
Zhang, K., Yang, Z., & Başar, T. (2019). Multi-agent reinforcement learning: A selective overview of theories and algorithms. arXiv preprint arXiv:1911.10635.

Vlontzos, A., Alansary, A., Kamnitsas, K., Rueckert, D., & Kainz, B. (2019, October). Multiple landmark detection using multi-agent reinforcement learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 262-270). Springer, Cham.

Multi-Agent Reinforcement Learning (II)

- **MARL can be placed into three groups, cooperative, competitive, and a mix of the two**
  - Cooperative agents
    - Agents collaborate to optimize a common long-term return, ex) Communication network
  - Competitive agents
    - The return of agents usually sum up to zero, ex) Two-player games including Go
  - Mix of cooperative and competitive agents:
    - Involving both agents with general-sum returns, ex) DOTA





Cooperative agents(multi-walker)

Competitive agents(pong, boxing)

Mixed agents(MAgent Battle)

Zhang, K., Yang, Z., & Başar, T. (2019). Multi-agent reinforcement learning: A selective overview of theories and algorithms. arXiv preprint arXiv:1911.10635.

PettingZoo environment

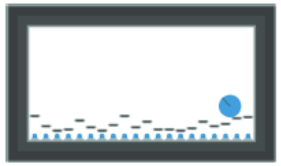
- **A library of diverse sets of multi-agent environment under a python API, akin to what OpenAI's Gym library**
  - PettingZoo's API is very similar to Gym's API and can be immediately understood by novices
  - PettingZoo provide many default environments within one package
    - Multi-agent fork of the Arcade Learning Environment(Atari), Multi-Agent Particle Environments(MPE), MAgent, SISL
    - 또한, 개발자들이 pettingZoo API에 맞게 자신의 환경을 개발할 수 있는 방법에 대해서도 자신들의 사이트에 소개함

```python
import gym
env = gym.make('CartPole-v0')
observation = env.reset()
for _ in range(1000):
    env.render()
    action = policy(observation)
    observation, reward, done, info = env.step(action)
env.close()
```

```python
from pettingzoo.butterfly import pistonball_v0
env = pistonball_v0.env()
env.reset()
for agent in env.agent_iter(1000):
    env.render()
    observation, reward, done, info = env.last()
    action = policy(observation, agent)
    env.step(action)
env.close()
```

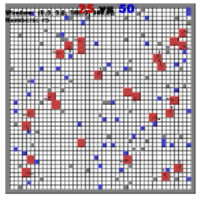
Basic Gym's API and basic PettingZoo's API

(a) Atari: Space Invaders



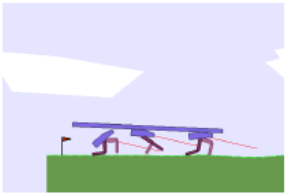
(b) Butterfly: Pistonball

(c) Classic: Chess



(d) MAgent: Adversarial Pursuit

(e) MPE: Simple Adversary



(f) SISL: Multiwalker

Diverse default environments

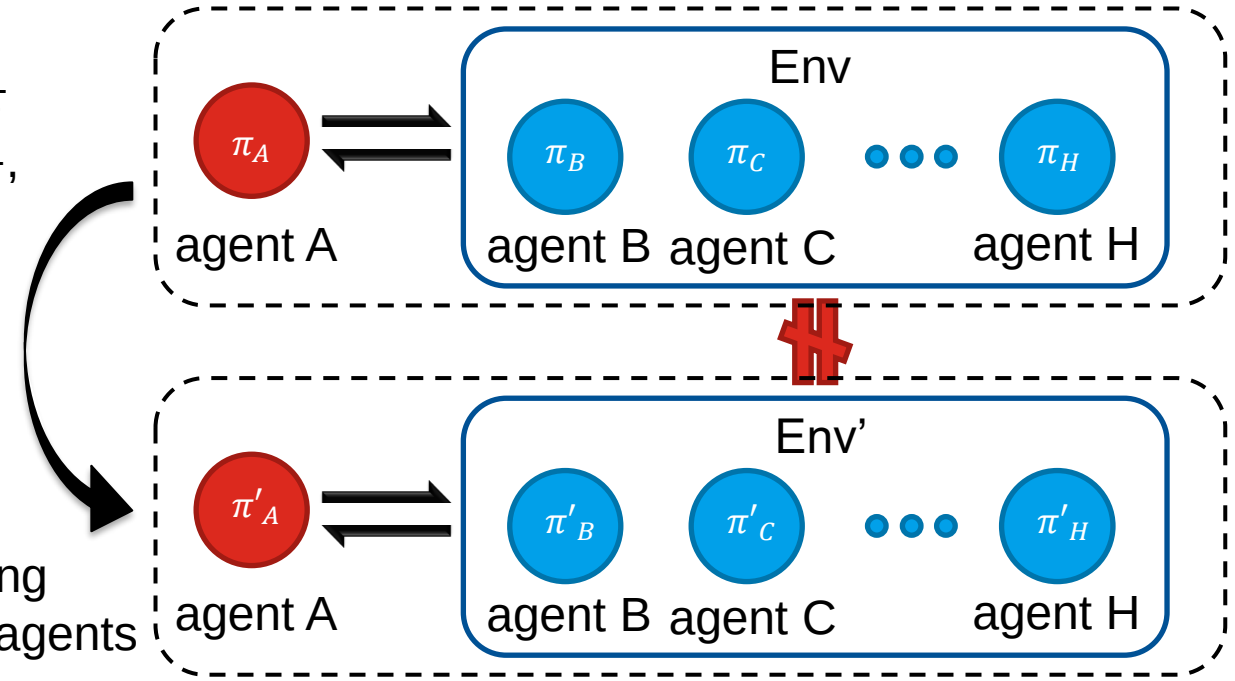
Terry, J. K., Black, B., Hari, A., Santos, L., Dieffendahl, C., Williams, N. L., ... & Ravi, P. (2020). PettingZoo: Gym for Multi-Agent Reinforcement Learning. arXiv preprint arXiv:2009.14471.

Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments (MADDPG)

Limitation of traditional actor-critic method in multi-agent domain

- **Value based algorithms have challenged by an inherent non-stationarity of the environment in multi-agent domain**
  - Each agent's policy is changing as training progresses, and the environment becomes non-stationary from the perspective of any individual agent

- **Policy-gradients algorithms have suffered from a variance that increases as the number of agents grows.**
  - PG 계열 알고리즘들은 off-policy 학습이 가능한 Value 계열 알고리즘에 비해 샘플 효율이 낮기 때문
  - *Off-policy 기법을 PG계열 알고리즘에 적용할 경우, importance sampling등 비용이 비싼 연산이 필요



https://blog.naver.com/PostView.nhn?blogId=jk96491&logNo=222271152952&categoryNo=27&parentCategoryNo=0&viewDate=&currentPage=1&postListTopCurrentPage=&from=postList&userTopListOpen=true&userTopListCount=5&userTopListManageOpen=false&userTopListCurrentPage=1

Background

- **Markov Games**
  - Multi-agent extension of Markov decision processes(MDPs), $< \boldsymbol{S}, A_1, A_2, \ldots, A_N, , \boldsymbol{O}, \boldsymbol{T}, R_1, R_2, \ldots R_N >$
    - N: Number of agent
    - $\boldsymbol{S}$: Set of states
    - A_i: Set of actions for agent i
    - $\boldsymbol{O}$: Set of observations
    - $\boldsymbol{T}$: Transition function, $\boldsymbol{S} \times A_1 \times \ldots A_N \times \boldsymbol{O} \rightarrow \boldsymbol{S}$
    - R_i: Reward for agent i, $\boldsymbol{S} \times A_1 \times \ldots A_N \times O \rightarrow \mathbb{R}$

- **Deterministic Policy Gradient(DPG) algorithms.**
  - Deterministic policy $\mu_\theta : S \rightarrow A$, 주어진 상태변수 값에서 행동변수의 값이 확정적으로 계산되는 정책
  - Objective function $J(\theta) = \mathbb{E}_{s \sim p^{\mu_\theta}}[R]$, 정책을 θ로 parameter화 할 경우, θ에 따른 강화 학습의 목적 함수
  - Deterministic policy gradient $\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}}[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s, a)|_{a = \mu_\theta(s)}]$
    - DPG의 경우, importance sampling 없이 off-policy learning이 가능

- **Deep Deterministic Policy Gradient(DDPG) algorithms.**
  - DPG 알고리즘의 policy μ와 critic Q^μ를 심층 신경망을 통해 근사한 알고리즘

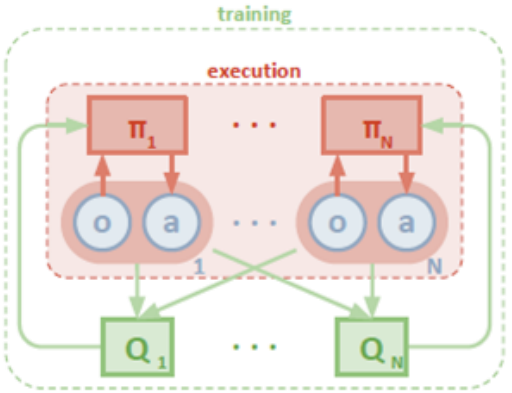
MADDPG (I)

- **Attributes of proposed multi-agent learning algorithm**
  - Leading to learned policies that only use local information (i.e. their own obs) at execution time
    - 학습시에는 모든 agent의 정보를 이용해 학습, 실행 시에는 자신의 정보만을 이용하여 행동을 선택
  - Doing not assume a differentiable model of the environment dynamics
    - 환경의 모델 P를 differentiable 하다고 가정하여, model을 학습에 이용하던 이전 연구와의 차이점을 강조
  - Doing not assume any certain structure on the communication method between agents
    - Agent사이 통신 구조를 특정한 구조로써 가정하지 않음. (통신이 무조건 가능하다고 가정한 것으로 파악됨)
  - Being applicable not only to cooperative interaction but to competitive or mixed interaction involving both physical and communicative behavior
    - 이러한 셋팅과, 뒤이어 설명할 아이디어들 덕분에, 협력적인 행위, 경쟁적인 행위, 혼합 행위 모두 학습이 가능한 알고리즘 즉, general-purpose multi-agent learning algorithm을 논문을 통해 제안하였다고 저자들은 기술

MADDPG (II)

- **Centralized Q func training with decentralized execution(each agent have centralized critic)**
  - Allowing the policies to use extra information to ease training
    - 일반적인 Q 함수는 학습, 실행 시 다른 정보를 포함하지 않음.

  - $Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, \ldots, a_N)$ is a ***centralized action-value function*** of agent i
    - $a_1, \ldots, a_N$ refers to actions of all agents, $\mathbf{x}$ refers to observation, $\mathbf{x} = (o_1, \ldots o_N)$
    - If Q_i^{μ} taken actions of all agents, the env become stationary even as policies change
      - Since $P(s'|s, a_1, \ldots, a_N, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N) = P(s'|s, a_1, \ldots a_N) = P(s'|s, a_1, \ldots, a_N, \boldsymbol{\mu}'_1, \ldots, \boldsymbol{\mu}'_N) for\ any\ \boldsymbol{\mu}_i \neq \boldsymbol{\mu}'_i$
    - $\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}\left[\left(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y_i\right)^2\right],\ \ y_i = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a'_1, \ldots, a'_N)|_{a'_j = \boldsymbol{\mu}'_j(o_j)}$



Overview of decentralized actor, centralized critic approach

  - Considering a game with N agents with deterministic policies with N policies $\boldsymbol{\mu}_{\theta_i}$ w.r.t parameters θ_i
    - The gradient can be written as:

$$\nabla_{\theta_i} J(\mu_{\theta_i}) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}[\nabla_\theta \mu_{\theta_i}(a_i|o_i) \nabla_{a_i} Q_i^{\mu}(x, a_1, \ldots, a_N)|a_i = \mu_{\theta_i}(o_i)]$$

MADDPG (III)

- **Inferring policies of other agents to remove the assumption of knowing other agents' policies**
  - 제안된 알고리즘의 centralized Q를 학습하기 위한 loss식:
    - $\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}\left[\left(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y_i\right)^2\right], \;\; y_i = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N')|_{a_j' = \boldsymbol{\mu}_j'(o_j)}$
  - 이 식에 따르면, i번째 agent의 target Q인 y_i를 학습하기 위해 他 agent의 정책μ_j가 필요.
    - 실제 세상에서 다른 agent의 정책을 안다는 가정을 하기 힘든 상황들이 존재할 수 있음.
    - 저자들은 그럴 경우에 대한 보완책으로써, 다른 agent의 정책들을 自 agent가 추론할 수 있는 방안에 대해 설명
  - Each agent i can additionally maintain an approximation $\hat{\mu}_{\phi_i^j}$ to true policy of agent j.
    - This approximate policy is learned by maximizing log probability of agent j's actions, with an entropy regularizer:
      $$L(\phi_i^j) = -E_{o_j,a_j}[\log \hat{\boldsymbol{\mu}}_i^j(a_j|o_j) + \lambda H(\hat{\boldsymbol{\mu}}_i^j)$$
    - With the approximate policies, target Q, y can be replaced by an approximate value $\hat{y}$ calculated as follows:
      $$\hat{y_i} = r_i + \gamma Q_i^{\mu'}\left(\mathbf{x}', \hat{\boldsymbol{\mu}}'^1_i(o_1), \ldots, \boldsymbol{\mu}_i'(o_i), \ldots, \hat{\boldsymbol{\mu}}'^N_i(o_N)\right)$$
    - Centralized Q function $Q_i^{\boldsymbol{\mu}}$ 업데이트 전에 agent i가 추론한 j 번째 agent의 approximate policy $\hat{\boldsymbol{\mu}}_{\phi_i^j}$들의 업데이트 수행

MADDPG (IV)

- **Using ensemble of policies for training agents to improve the stability of multi-agent policies**
  - K개의 sub-policy들을 무작위로 선택하여 학습해 나가며, multi-agent들의 학습을 보다 안정적으로 수행
    - 이는 특히, 경쟁적인 행위를 학습하는 경우를 위해 제안 되었으며, 경쟁적인 행위를 학습 시, 그들의 경쟁자 (competitors)들에 대해 정책이 오버 피팅되는 것을 방지하기 위한 방법이라고 설명
  - Policy $\boldsymbol{\mu}_i$ is an ensemble of K different sub-policies with sub-policy k denoted by $\boldsymbol{\mu}_{\theta_i^{(k)}}$
    - For agent i, the ensemble objective $J_e(\boldsymbol{\mu}_i)$ is as follows:

$$J_e(\boldsymbol{\mu}_i) = \mathbb{E}_{k\sim unif(1,K), s\sim p^{\boldsymbol{\mu}}, a\sim \boldsymbol{\mu}_i^{(k)}}[R_i]$$

  - 이러한 sub-policy들은 에피소드 진행 시 하나가 선택되어 rollout을 수행하며, agent i의 replay buffer $D_i^{(k)}$ 또한 sub-policy의 개수 만큼 인스턴스화 하여 사용
    - Accordingly, the gradient of the ensemble objective with respect to $\theta_i^{(k)}$ as follows:

$$\nabla_{\theta_i^{(k)}} J_e(\boldsymbol{\mu}_i) = \frac{1}{K}\mathbb{E}_{x,a\sim D_i^{(k)}}\left[\nabla_{\theta_i^{(k)}} \boldsymbol{\mu}_i^{(k)}(a_i|o_i)\nabla_{a_i} Q^{\boldsymbol{\mu}_i}(\mathbf{x}, a_1, \ldots, a_N)|_{a_i = \boldsymbol{\mu}_i^{(k)}(o_i)}\right]$$

Experiments (I)

- **Environments(Multi-agent Particle Environment)**
  - Cooperative Communication**(Cooperative env)**
    - Speaker와 listener가 존재, 리스너는 스피커가 외치는 특정 색의 랜드마크로 이동해야 하며, 정확한 랜드마크까지의 거리를 기반으로 보상을 받는 환경
  - Predator-Prey**(Competitive env)**
    - N개의 느린 협력적인 predator agent가 더욱 빠른 prey agent를 추적해야 하며, L개의 임의의 랜드마크가 장애물로써 생성되는 환경. 포식자가 도망자를 잡을 경우, 협력적인 포식자들은 보상을, 적대적인 도망자는 페널티를 받는 환경
    - 에이전트들은 각 에이전트들의 위치와 속도, 랜드마크의 위치를 관측 가능
  - Cooperative navigation**(not cooperative, not competitive env)**
    - Agent들은 각기 다른 에이전트와 랜드마크까지의 상대 위치를 관측하며, 랜드마크 별 근접한 에이전트와의 거리를 기반으로 보상을 받는 환경
  - Physical Deception**(Competitive env)**
    - 해석은 하였으나 지면이 부족해 생략

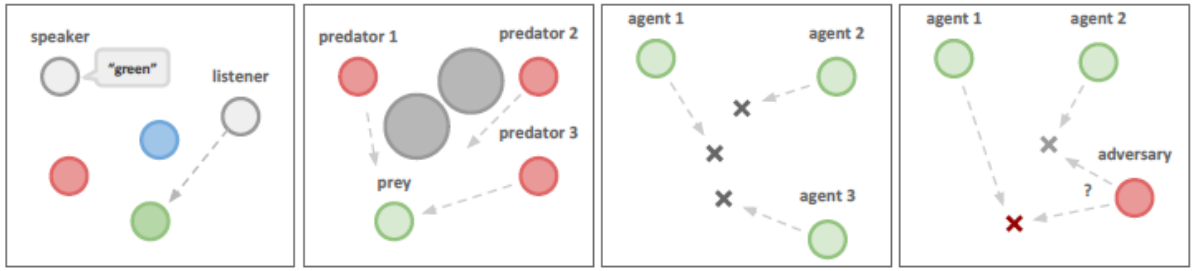


Figure 2: Illustrations of the experimental environment and some tasks we consider, including a) *Cooperative Communication* b) *Predator-Prey* c) *Cooperative Navigation* d) *Physical Deception*. See webpage for videos of all experimental results.

https://github.com/openai/multiagent-particle-envs

Experiments (II)

- **Evaluation**
  - Comparison to decentralized RL methods
    - TRPO, DDPG등 기존의 RL 기법들은 간단한 문제임에도 불구, 학습에 어려움이 있음을 보여줌
  - Effect of learning policies of other agents
    - Centralized Q function을 학습 함에 있어, 다른 agent의 정책을 완벽히 알지 못한채로, 제안한 approximation 방법을 적용할 경우 그렇지 않은 경우와 같은 결과를 얻었다는 것을 보여줌. (우측 그래프는 정책 사이 KL-divergence 값)
  - Effect of training with policy ensembles
    - Competitive한 환경 들에서, agent와 adversary의 policy를 single policy, ensemble policy로 적용해본 결과, agent가 ensemble policy를 가지고, adversary가 single policy를 가질 경우가 가장 좋은 성능을 보였음을 보여줌

Figure 6: Policy learning success rate on cooperative communication after 25000 episodes.

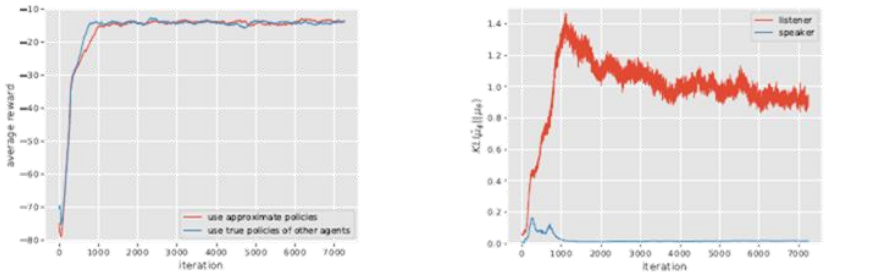


Figure 7: Effectiveness of learning by approximating policies of other agents in the cooperative communication scenario. *Left:* plot of the reward over number of iterations; MADDPG agents quickly learn to solve the task when approximating the policies of others. *Right:* KL divergence between the approximate policies and the true policies.

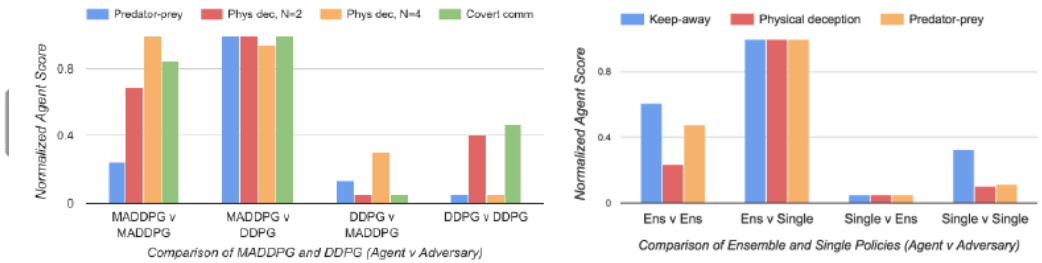


Figure 3: Comparison between MADDPG and DDPG (left), and between single policy MADDPG and ensemble MADDPG (right) on the competitive environments. Each bar cluster shows the 0-1 normalized score for a set of competing policies (agent v adversary), where a higher score is better for the agent. In all cases, MADDPG outperforms DDPG when directly pitted against it, and similarly for the ensemble against the single MADDPG policies. Full results are given in the Appendix.

Application to pettingZoo environment

Environments setting

- **Selected environments**
  - Multi-walker
    - 3개의 bi pedal walker가 위의 쟁반을 떨어뜨리지 않은 채 걸어가는 것을 목표로 하는 게임
    - Cooperative agents(3), scalar features, continuous action space
  - Prospector
    - 4명의 prospector agent들은 황금을 탐사하며, 한 번에 한 개의 금을 보유할 수 있으며, 보유한 금덩어리를 3개의 banker agent에게 전달하여 보상을 얻을 수 있음. Banker의 경우 prospector agent가 근처에 놓은 금근처로 이동하여 금을 입수하면 보상을 얻을 수 있음
    - Prospector agent들은 움직임과 회전이 가능하며, banker agent들은 이동만 가능
    - Competitive agents(4), (3), image features, continuous action space

Cooperative agents(multiwalker)

Mixed agents(prospector)

Implementation status

- **Env**
  - Multi-walker (O)
  - Prospector (X)

- **Current status**