MADDPG: Multi-Agent Deep Deterministic Policy Gradients:

https://arxiv.org/pdf/1706.02275.pdf

Paper Summary Notes

1 Main Ideas

Central Question of the Paper: *How can we create a general-purpose multi-agent extension of actor-critic methods?*

1.1 Why does traditional Rl fail?

- *Q learning*: environment becomes non-stationary from the perspective of any individual agents (in a way that is not explainable by changes in the agent's own policy), making Q-learning very unstable
- *Policy Gradients*: variance of gradients becomes extremely high when coordination of multiple agents is required

1.2 Goal: general purpose multi-agent reinforcement learning algorithm

1. leads to learned policies that only use local information at test time
2. does not assume differentiable model of the environment and does not assume any particular communication assumption between agents
3. is applicable to both cooperative, competitive and mixed environments

1.3 High level idea

- adopt framework of *centralized training with decentralized execution*, which allows policies to use extra information to ease training, so long as it's not necessary at test time
  - critic will have augmented information about other policies, actor remain local, acting in decentralized manner
  - since each agent stores its own generalized critic, this allows for different rewards for different agents
- *Inferred policies*: an extension that works when we don't have the assumption of knowing other agents policies. In this case we learn an approximate model of other agent policies, and adapt the same framework above
- *Ensemble policies*: improves stability of learning, by fitting an ensemble of K policies for each agent

2 Background

2.1 Partially Observable Markov Games

- assume N agents
- Global state $\mathcal{S}$ which contains information from all the agents
- Actions $\mathcal{A}_i$ and observation $\mathcal{O}_i$ for each agent $i; i = 1, ..., N$
- each agent acts according to their own stochastic policy:

$$\pi_{\theta_i} : \mathcal{O}_i \times \mathcal{A}_i \longrightarrow [0, 1] \tag{1}$$

  which produces the next global state according to the transition function:

$$\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times ... \times ...\mathcal{A}_N \longrightarrow \mathcal{S} \tag{2}$$

- Each agent receives rewards $r_i : \mathcal{S} \times \mathcal{A}_i \longrightarrow \mathbb{R}$ and tries to maximize their return:

$$R_i = \sum_{t=0}^{t=T} \gamma^t \cdot r_i^t \tag{3}$$

  over some horizon T with discount factor γ

2.2 Q learning

Make use of the action-value function:

$$Q^\pi(s, a) = \mathbb{E}[R|s^t = s, a^t = a] \tag{4}$$

and minimize the square loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(Q^*(s, a, |\theta) - y)^2] \tag{5}$$

$$y = r + \gamma max_{a'} Q^*(s', a') \tag{6}$$

by sampling transitions (s, a, r, s') from a replay buffer $\mathcal{D}$

- although it can be applied to each agent independently, the environment appears non-stationary, which makes the convergence very unstable

2.3 Policy Gradients

Directly ascent on the objective

$$J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[R] \tag{7}$$

by taking steps according to $\nabla_\theta J(\theta)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta log\pi(a|s) Q^\pi(s, a)] \tag{8}$$

- This gradient estimate is known to have high variance, which becomes exacerbated in mulit-agent settings, since an agent's reward will depend on actions of many other agents, not accounted for in the local observation

3 Methods

3.1 Multi-Agent Actor Critic

3.1.1 Centralized Training with Decentralized Execution

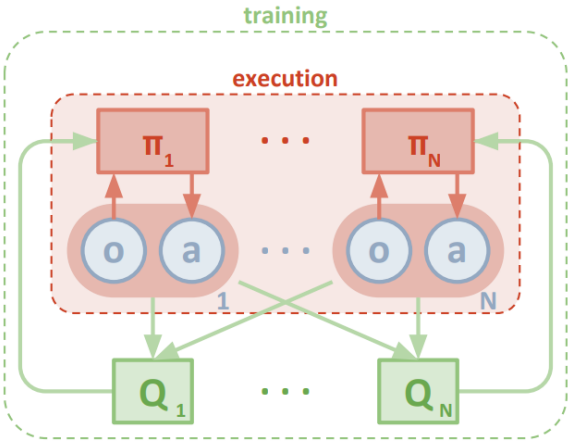
Intuition: we can make the environment stationary with respect to the other policies if we knew the actions taken by all the agents

- propose a simple extension which gives the critic extra information about the policies of other agents, leading to the *augmented state-value function*:

$$Q_i^{\pi}(x, a_1, a_2, ..., a_N) \tag{9}$$

  where the input x contains observation of all agents $x = (o_1, .., o_N)$ along with potentially more information

- The policy gradient actor-critic extends naturally for each agent i using the augmented state-action value instead of the normal one



3.1.2 Pseudo-Code

Multi-Agent Deep Deterministic Policy Gradient Algorithm

For completeness, we provide the MADDPG algorithm below.

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

3.2 Inferring Policies

What if we don't know the observation of other agents?

Intuition: let each agent maintain an approximation of the other agent's policies, use this in our augmented state-value input x

- Each agent i maintain an approximation $\mu_{\phi_i^j}$ of agent j policy with parameters ϕ that we fit using max-likelihood with entropy regularized:

$$\mathcal{L}(\phi_i^j) = -\mathbb{E}_{o_j, a_j}[log\mu_i^j(a_j, o_j) + \lambda\mathcal{H}(\mu_i^j)] \tag{10}$$

With approximate policies, the target value y can be replaced with the estimated policies, and optimized online.

3.3 Ensemble Policies

Intuition: we can reduce over-fitting in adverserial setting by maintaining an ensemble of K policies for each agent

- For each agent, train K different sub-policies, and at each episode uniformly sample one of these, and execute accordingly

4 Experiments

4.1 Environments

- *Cooperative communication*: : 2 cooperative agents, speaker and listener, placed in env with three landmarks of different colors. Speaker tells listener which landmark to navigate to, listener rewards based on distance to correct landmark, speaker can communicate at each time step which is observed by listener
- *Cooperative navigation*: cooperative agents must cover all of the landmarks, and thus communicate which ones they plan on going to, and are penalized when colliding with each other
- *Keep away*: L landmarks including a target, N cooperating agents who know the target landmark and are rewarded based on distance to target, and M adversarial agents who must prevent cooperating agents from reaching the target, which they do not directly have knowledge of. They must infer from movement of the cooperating agents
- *Physical Deception*: N agents cooperate to reach a single landmark out of N. Only one of the agents needs to get to the landmark, and there is lone adversary which also desires to reach the target, but it does not know a priori which is the target. Cooperating agents should learn to move around together and spread out so that the adversary does not know which is the true target
- *Predator-prey*: N slow cooperating agents must chase a faster adversary
- *Covert communication*: cryptographic env where Alice must communicate message to Bob, who must reconstruct the message. An adversary Eve is also observing the channel, and wants to reconstruct the message, Alice and Bob penalized based on eves reconstruction and thus Alice must encode her message u e randomly generated key, only known to Alice and bob

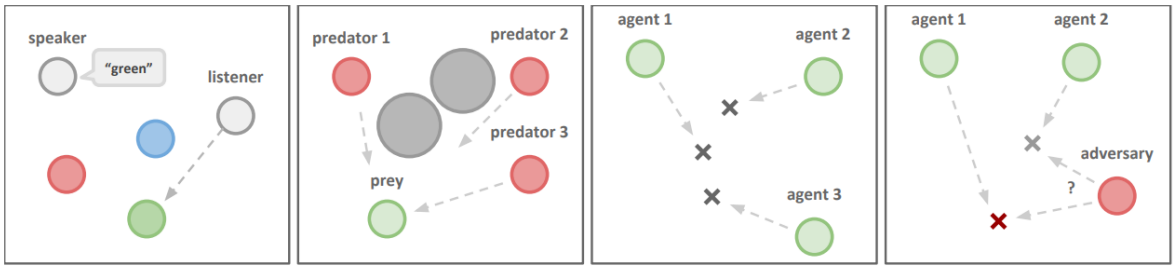
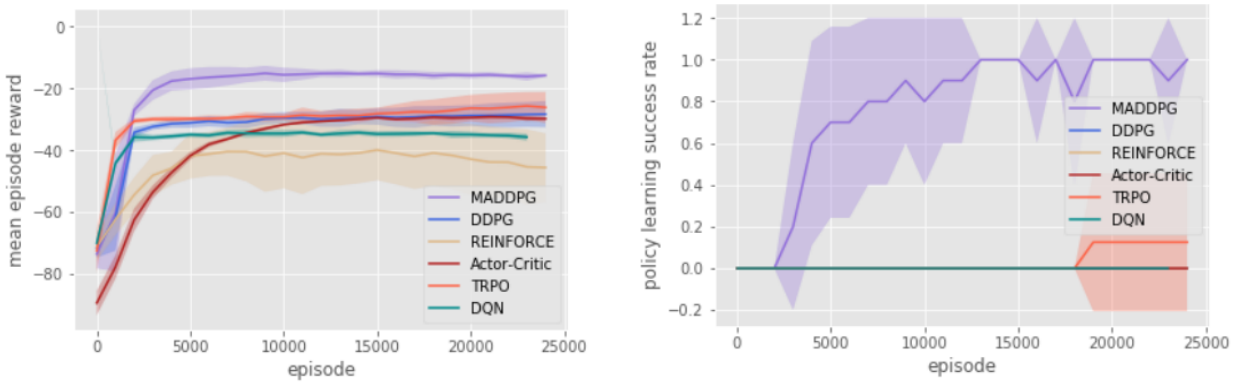


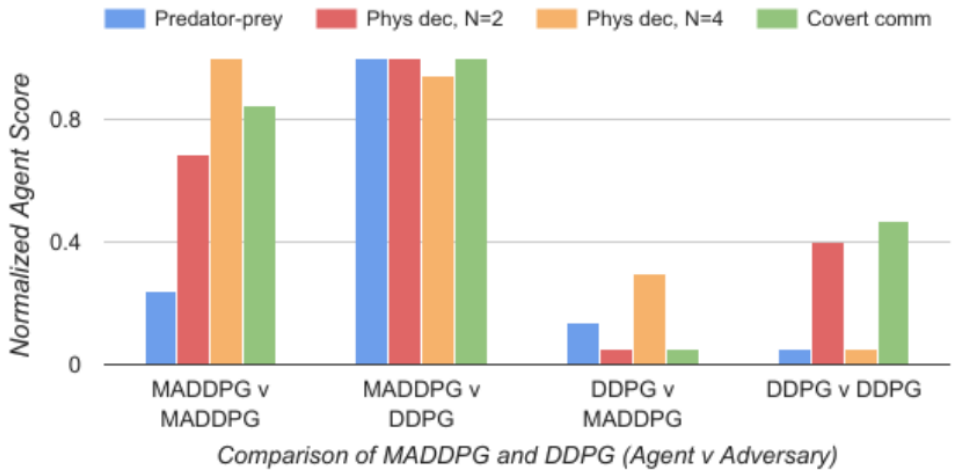
Figure 2: Illustrations of the experimental environment and some tasks we consider, including a) *Cooperative Communication* b) *Predator-Prey* c) *Cooperative Navigation* d) *Physical Deception*. See webpage for videos of all experimental results.

4.2 Results

4.2.1 Maddpg vs DDPG

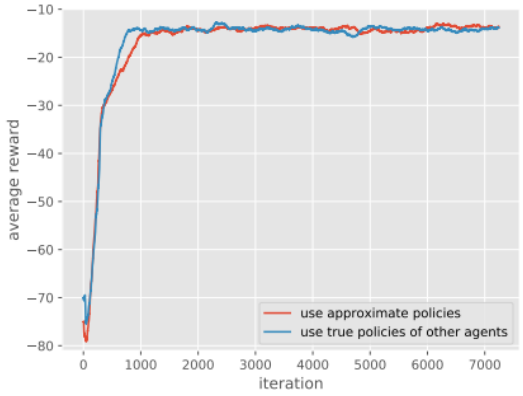


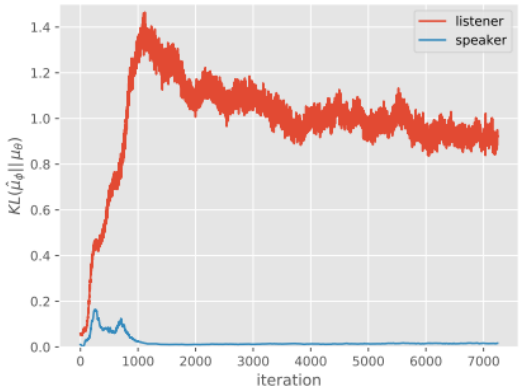
- we see the multi-agent version is the only own able to learn the multi-agent settings, likely because in traditional rl, we have inconssistency gradient signal
  - example: if the speaker utters correct symbol, but listener moves in wrong direction, the speaker is penalized



- bar group 2: shows how multi-agent cooperating policies are able to completely outperform adversaries that are trained independently
- bar group 3: shows how independently trained policies do very poorly against adversaries that are trained using MADDPG

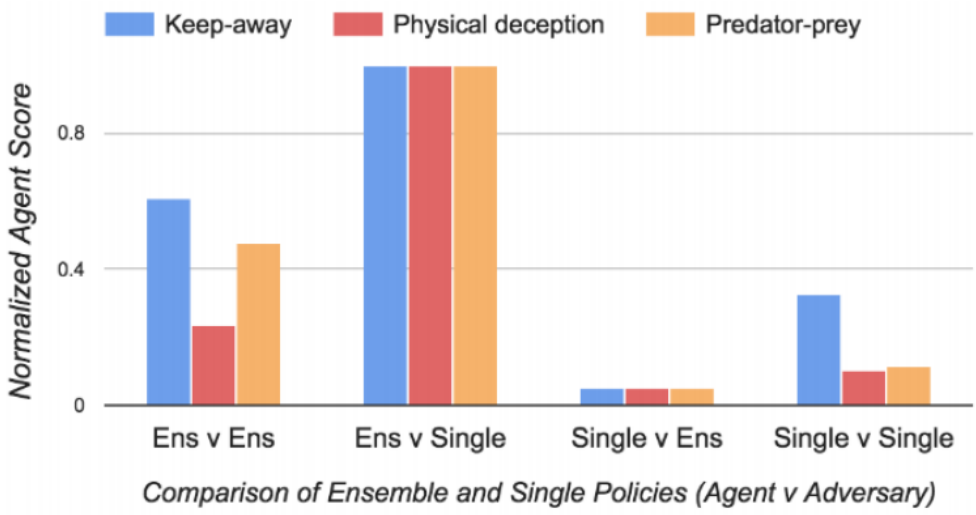
4.2.2 Inferring Policies





- under the setup of inferring policies, despite not perfectly fitting, we are able to achieve the same success rate!

4.2.3 Ensemble Policies



Ensemble policies make it difficult to exploit a single behaviour, which results in:

- bar group 2: shows how ensemble cooperating policies are able to completely outperform adversaries that are trained in single policy
- bar group 3: shows how single trained policies do very poorly against adversaries that are trained using ensemble