2019 CS420, Machine Learning, Lecture 14

Multi-Agent Reinforcement Learning

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

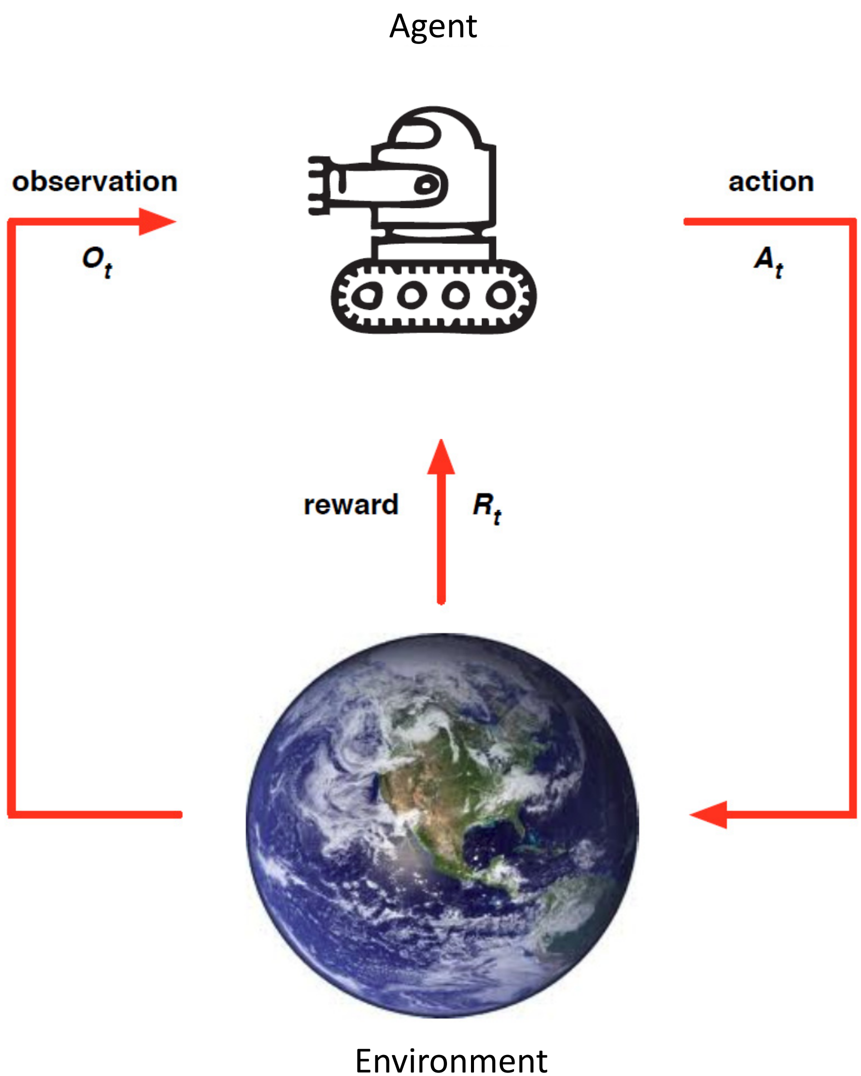
http://wnzhang.net/teaching/cs420/index.html

Reinforcement Learning

- Learning from interaction with the environment

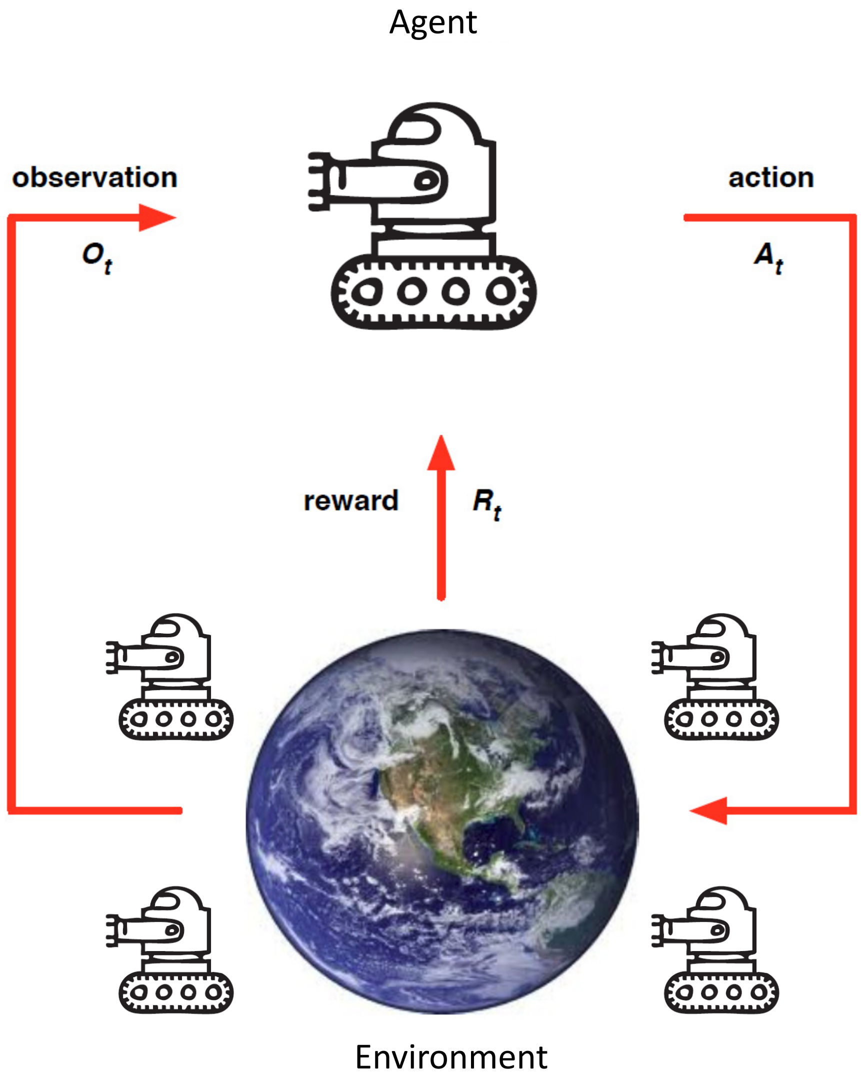
- The agent
  - senses the observations from environment
  - takes actions to deliver to the environment
  - gets reward signal from the environment

- Normally, the environment is stationary

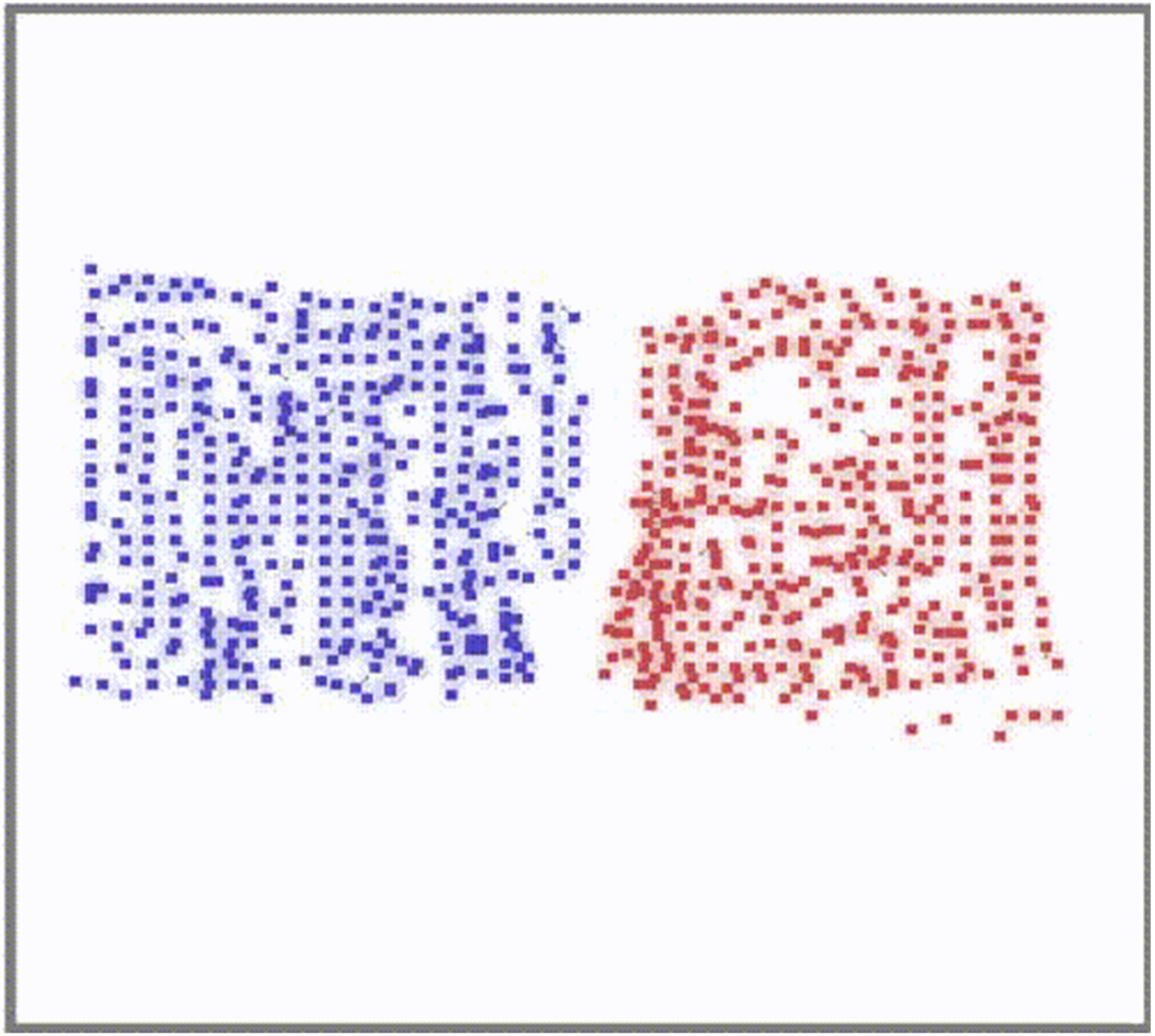


Multi-Agent Reinforcement Learning

- Learning from interaction with the environment
- The environment contains other agents that are learning and updating
- Non-stationary environment



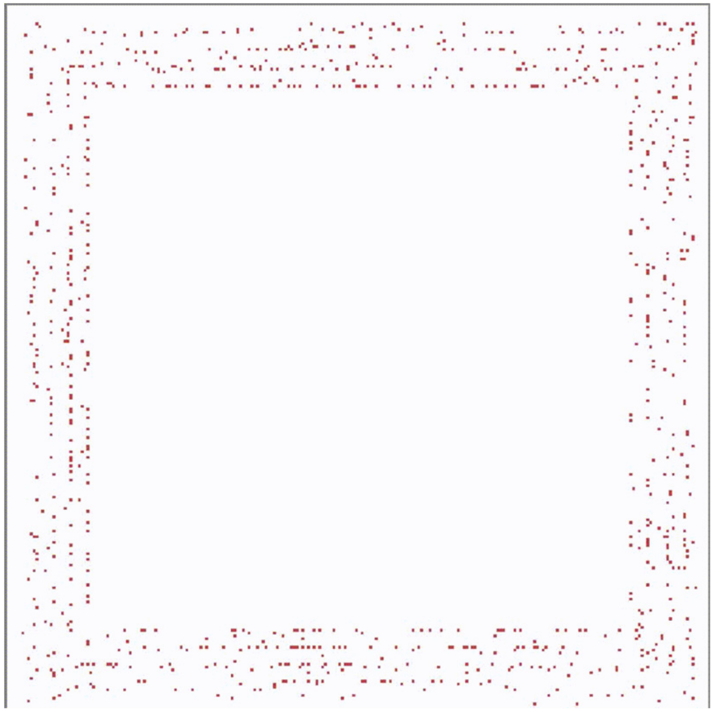
Case 1: Battle Game

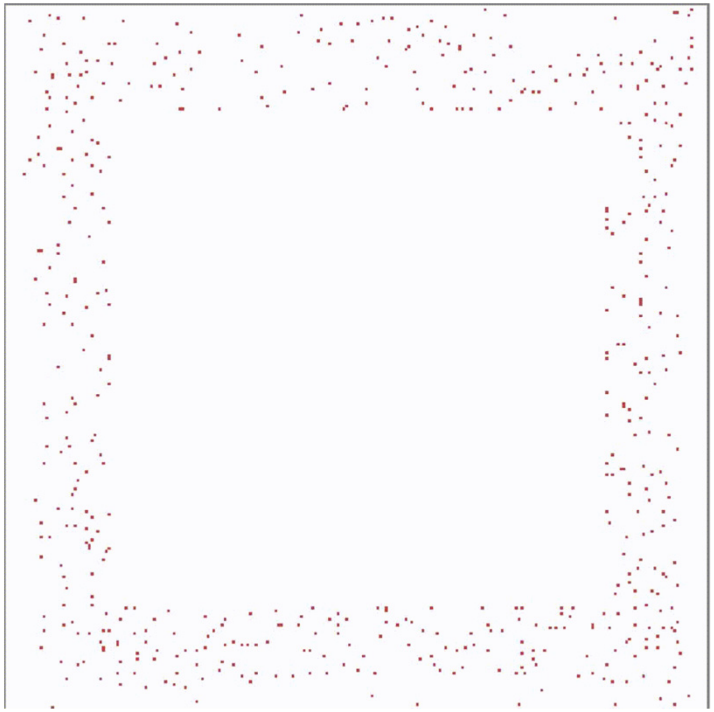


Lianmin Zheng, Weinan Zhang et al. Magent: a many-agent reinforcement learning platform for artificial collective intelligence. NIPS17 & AAAI18.

Case 2: Army Align

- Let an army of agents align a particular pattern





Lianmin Zheng, Weinan Zhang et al. Magent: a many-agent reinforcement learning platform for artificial collective intelligence. NIPS17 & AAAI18.

Case 3: Decentralized Game AI

- Designing multi-agent communications and co-learning algorithms for elaborate collective game intelligence

RTS Games

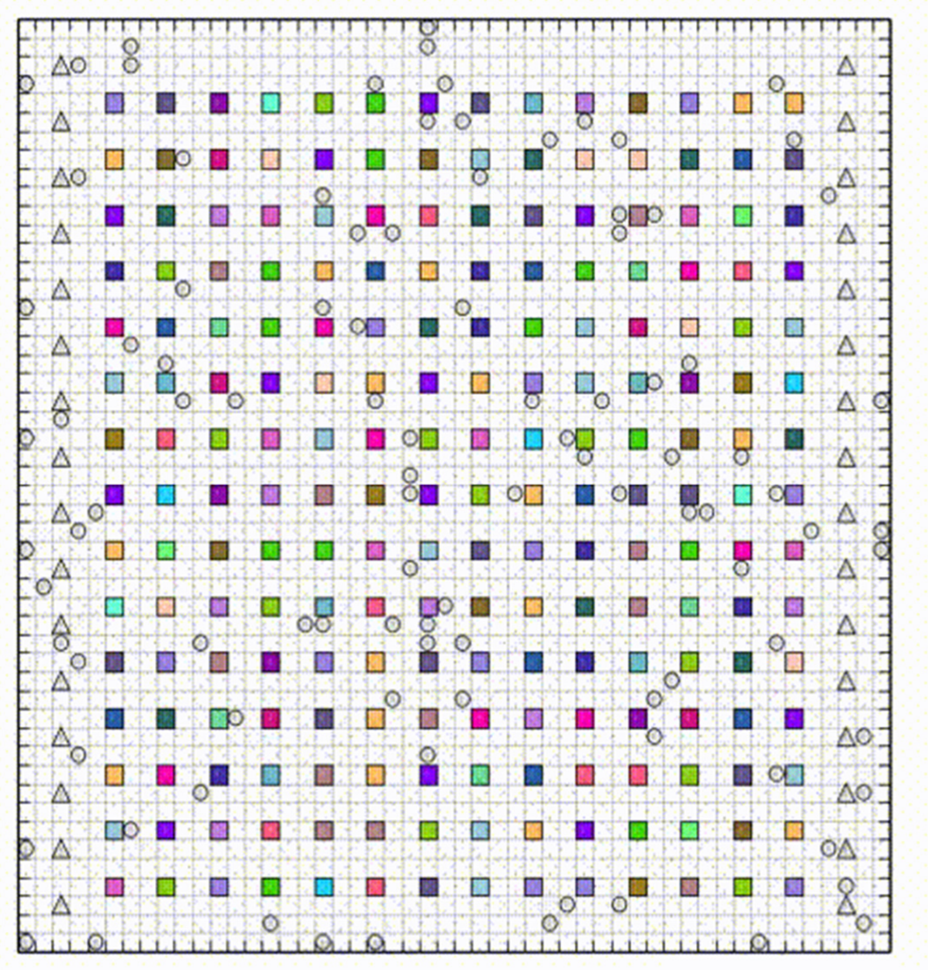
MOBA Games

Peng, Peng, et al. "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games." *NIPS workshop 2017.*

Case 4: City Brain Simulation

- Designing
  - Car routing policy
  - Traffic light controller
  - Fleet management & texi dispatch

Case 5: Storage Sorting Robots



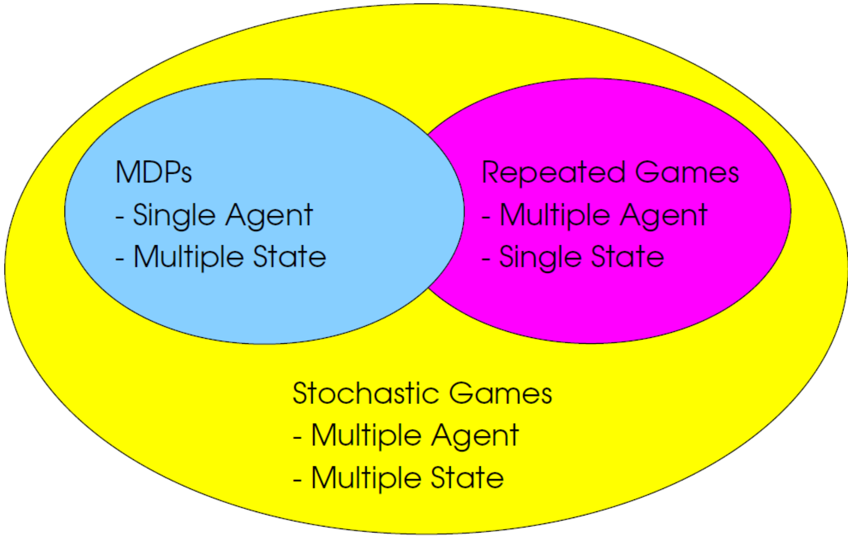
Difficulty in Multi-Agent Learning

- MAL is fundamentally more difficult
  - since agents not only interact with the environment but also with each other

- If use single-agent Q learning by considering other agents as a part of the environment
  - Such a setting breaks the theoretical convergence guarantees and makes the learning unstable
  - i.e., the changes in strategy of one agent would affect the strategies of other agents and vice versa

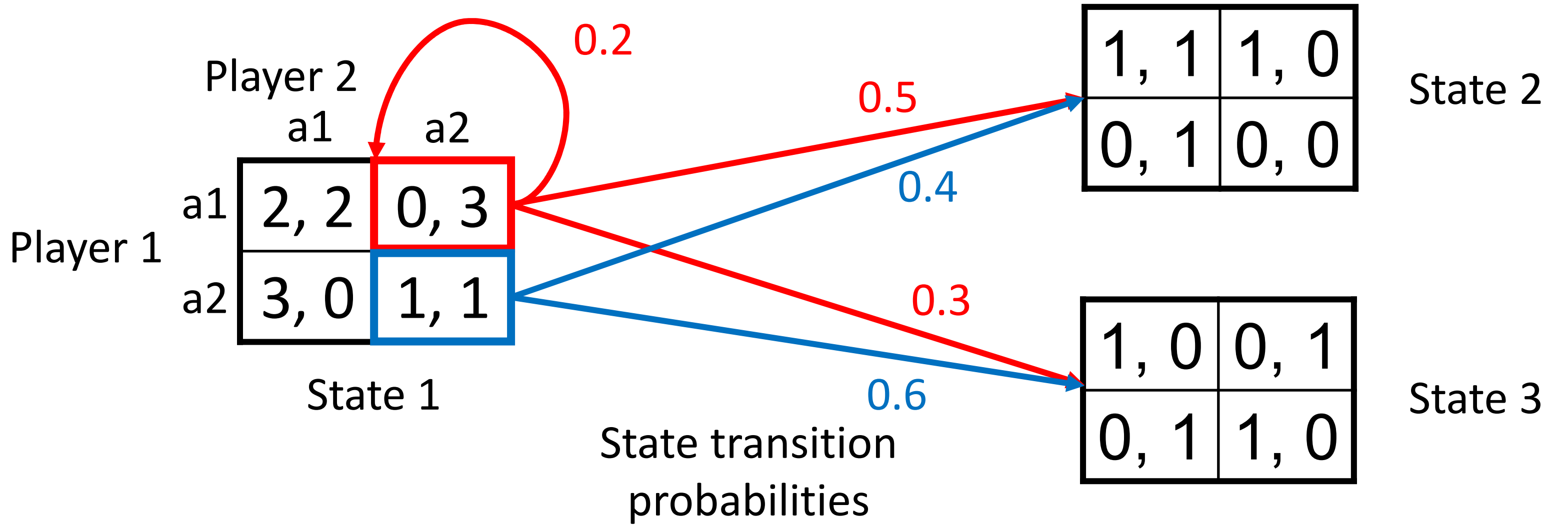
Sequential Decision Making

- 3 types of setting
  - Markov decision processes
    - one decision maker
    - multiple states
  - Repeated games
    - multiple decision makers
    - one state (e.g., one normal form game)
  - Stochastic games (Markov games)
    - multiple decision makers
    - multiple states (e.g., multiple normal form games)



Stochastic Games

- A stochastic game has multiple states and multiple agents
  - Each state corresponds to a normal-form game
  - After a round, the game randomly transits to another state
  - Transition probabilities depend on state and joint actions taken by all agents
- Typically rewards are discounted over time



Shapley, Lloyd S. "Stochastic games." *Proceedings of the national academy of sciences* 39.10 (1953): 1095-1100.

Definition of Stochastic Games

- A stochastic game is defined by

$$(\mathcal{S}, \mathcal{A}^1, \dots, \mathcal{A}^N, r^1, \dots, r^N, p, \gamma)$$

- State space: $\mathcal{S}$
- Action space of agent *j*: $\mathcal{A}^j, \;\; j \in \{1, \dots, N\}$
- Reward function of agent $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$
- Transition probability $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \Omega(\mathcal{S})$

The collection of probability distributions over *S*

- Discount factor across time $\gamma \in [0, 1)$

Policies in Stochastic Games

- For agent *j*, the corresponding policy is

$$\pi^j : \mathcal{S} \to \Omega(\mathcal{A}^j)$$

The collection of probability distributions over A^j

- The joint policy of all agents is $\boldsymbol{\pi} \triangleq [\pi^1, \ldots, \pi^N]$

- State value function of agent *j*

$$v^j_{\boldsymbol{\pi}}(s) = v^j(s; \boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\boldsymbol{\pi},p}\big[r^j_t | s_0 = s, \boldsymbol{\pi}\big].$$

- Action value function of agent *j* $Q^j_{\boldsymbol{\pi}} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$

$$Q^j_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r^j(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s' \sim p}[v^j_{\boldsymbol{\pi}}(s')]$$

$$[a^1, \ldots, a^N]$$

Independent Learning in SG

- For each agent *j*, assume the other agents' policies are stationary, thus the environment for *j* is stationary to perform Q-learning

$$Q(s, a^j, a^{-j}) \leftarrow Q(s, a^j, a^{-j}) + \alpha(r + \gamma \max_{a^{j\prime}} Q(s', a^{j\prime}, a^{-j\prime}) - Q(s, a^j, a^{-j}))$$

- Unfortunately, in SG with MARL, every agent is learning and updating its policy, making the environment non-stationary

Nash Equilibrium in SG

$$v^j_{\boldsymbol{\pi}}(s) = v^j(s;\boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\boldsymbol{\pi},p}[r^j_t | s_0 = s, \boldsymbol{\pi}]$$

- Optimizing $v^j_{\boldsymbol{\pi}}(s)$ for agent *j* depends on the joint policy $\boldsymbol{\pi}$

- Nash equilibrium in SG is represented by a particular joint policy

$$\boldsymbol{\pi}_* \triangleq [\pi^1_*, \ldots, \pi^N_*]$$

  such that nobody would like to change his policy given the others'

$$v^j(s;\boldsymbol{\pi}_*) = v^j(s;\pi^j_*, \boldsymbol{\pi}^{-j}_*) \geq v^j(s;\pi^j, \boldsymbol{\pi}^{-j}_*)$$

$$\boldsymbol{\pi}^{-j}_* \triangleq [\pi^1_*, \ldots, \pi^{j-1}_*, \pi^{j+1}_*, \ldots, \pi^N_*]$$

Nash Q-learning

- Given a Nash policy $\boldsymbol{\pi}_*$, the Nash value function

$$\boldsymbol{v}^{\texttt{Nash}}(s) \triangleq [v^1_{\boldsymbol{\pi}_*}(s), \ldots, v^N_{\boldsymbol{\pi}_*}(s)]$$

- Nash Q-learning defines an iterative procedure
    1. Solving the Nash equilibrium $\boldsymbol{\pi}_*$ of the current stage defined by $\{\boldsymbol{Q}_t\}$
    2. Improving the estimation of the Q-function with the new Nash value $\boldsymbol{v}^{\text{Nash}}$
- But Nash Q-learning suffers from
    - Very high computational complexity
    - May not work when other agents' policy is unavailable

From Multi- to Many-Agent RL

- What will happen when agent number grows?
  - Reward function of agent $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$
  - Transition probability $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \Omega(\mathcal{S})$
- Both reward function and state transition probability get exponentially larger
  - More difficult to model
  - The environment is more dynamic and sensitive
  - Need more exploration data
  - More computational resources

Idea: Taking Other Agents as A Whole

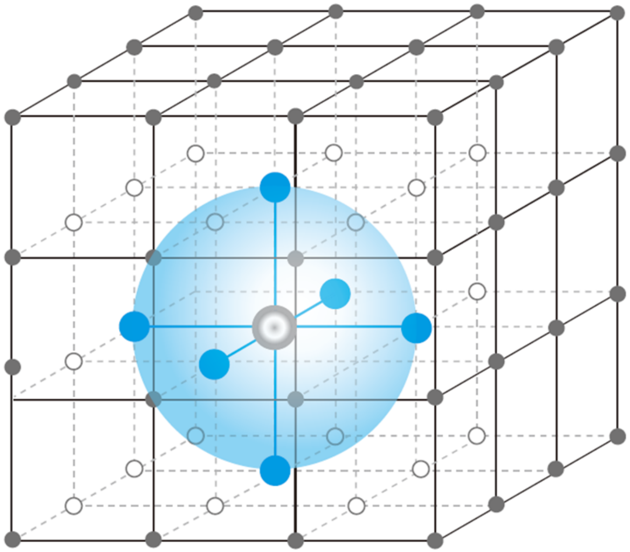
- In some many-body systems, the interaction between an agent and others can be approximated as that between the agent and the "mean agent" of others

Mean Field Multi-Agent RL

- Mean field approximation
  - Approximate the joint action value by factorizing the Q-function into pairwise interactions

$$Q^j(s, \boldsymbol{a}) = \frac{1}{N^j} \sum_{k \in \mathcal{N}(j)} Q^j(s, a^j, a^k)$$

Neighboring agent set of *j*



  - Significantly reduces the global interactions among agents
  - Still preserves global interactions of any agent pair

Yaodong Yang, Weinan Zhang et al. Mean Field Multi-Agent Reinforcement Learning. ICML 2018.

Action Representation

$$Q^j(s, \boldsymbol{a}) = \frac{1}{N^j} \sum_{k \in \mathcal{N}(j)} Q^j(s, a^j, a^k)$$

- Consider discrete action space
  - Action a^j of agent j is one-hot encoded as

    $$a^j \triangleq [a^j_1, \ldots, a^j_D]$$ Only one element is 1
  - The mean action based on the neighborhood of j is

    $$\bar{a}^j = \frac{1}{N^j} \sum_k a^k$$
  - Thus the action a^k of each neighbor k can be represented as

$$\underbrace{a^k = \bar{a}^j}_{\text{mean action}} + \underbrace{\delta a^{j,k}}_{\text{residual}}$$

$$\frac{1}{N^j} \sum_k a^{j,k} = 0$$

Residual sum is 0

Mean Field Approximation

- A 2-order Taylor expansion on Q-function

$$a^k = \bar{a}^j + \delta a^{j,k}$$

$$Q^j(s, \boldsymbol{a}) = \frac{1}{N^j} \sum_k Q^j(s, a^j, a^k)$$

$$= \frac{1}{N^j} \sum_k \left[Q^j(s, a^j, \bar{a}^j) + \nabla_{\bar{a}^j} Q^j(s, a^j, \bar{a}^j) \cdot \delta a^{j,k} + \frac{1}{2} \delta a^{j,k} \cdot \nabla^2_{\tilde{a}^{j,k}} Q^j(s, a^j, \tilde{a}^{j,k}) \cdot \delta a^{j,k} \right]$$

$$= Q^j(s, a^j, \bar{a}^j) + \nabla_{\bar{a}^j} Q^j(s, a^j, \bar{a}^j) \cdot \frac{1}{N^j} \sum_k \delta a^{j,k} + \frac{1}{2N^j} \sum_k \delta a^{j,k} \cdot \nabla^2_{\tilde{a}^{j,k}} Q^j(s, a^j, \tilde{a}^{j,k}) \cdot \delta a^{j,k}$$

$$= Q^j(s, a^j, \bar{a}^j) + \frac{1}{2N^j} \sum_k R^j_{s,a^j}(a^k)$$

$$\approx Q^j(s, a^j, \bar{a}^j)$$

External random signal for agent *j*

$$R^j_{s,a^j}(a^k) \triangleq \delta a^{j,k} \cdot \nabla^2_{\tilde{a}^{j,k}} Q^j(s, a^j, \tilde{a}^{j,k}) \cdot \delta a^{j,k}$$

$$\tilde{a}^{j,k} = \bar{a}^j + \epsilon^{j,k} \delta a^{j,k}$$

Q-function model the interaction between the agent's action and the mean action

Mean Field Q-Learning

- A softmax MF-Q policy

$$\pi_t^j(a^j|s,\bar{a}^j) = \frac{\exp\left(\beta Q_t^j(s,a^j,\bar{a}^j)\right)}{\sum_{a^{j'}\in\mathcal{A}^j}\exp\left(\beta Q_t^j(s,a^{j'},\bar{a}^j)\right)}$$

- Given an experience $\langle s, \boldsymbol{a}, \boldsymbol{r}, s', \bar{\boldsymbol{a}}\rangle$ sampled from replay buffer
  - Sample the next action a_-^j from $Q_{\phi_-^j}$
  - Set $y^j = r^j + \gamma\, Q_{\phi_-^j}(s', a_-^j, \bar{a}^j)$
  - Update Q function with the loss function

$$\mathcal{L}(\phi^j) = \left(y^j - Q_{\phi^j}(s^j, a^j, \bar{a}^j)\right)^2$$

MF-Q Convergence

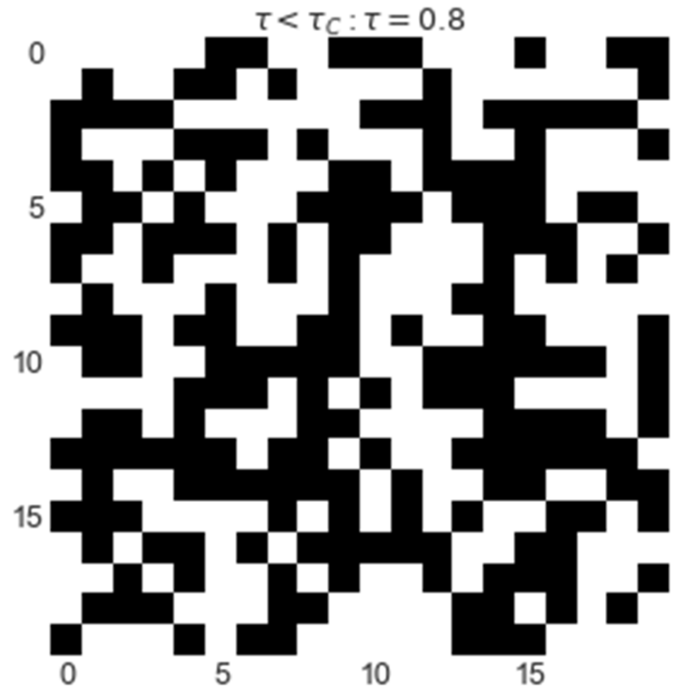
- Theorem: In a finite-state stochastic game, the Q values computed by the update rule of MF-Q converges to the Nash Q-value
  - under certain assumptions of reward function, policy form and game equilibrium

Experiment: Ising Model (IM)

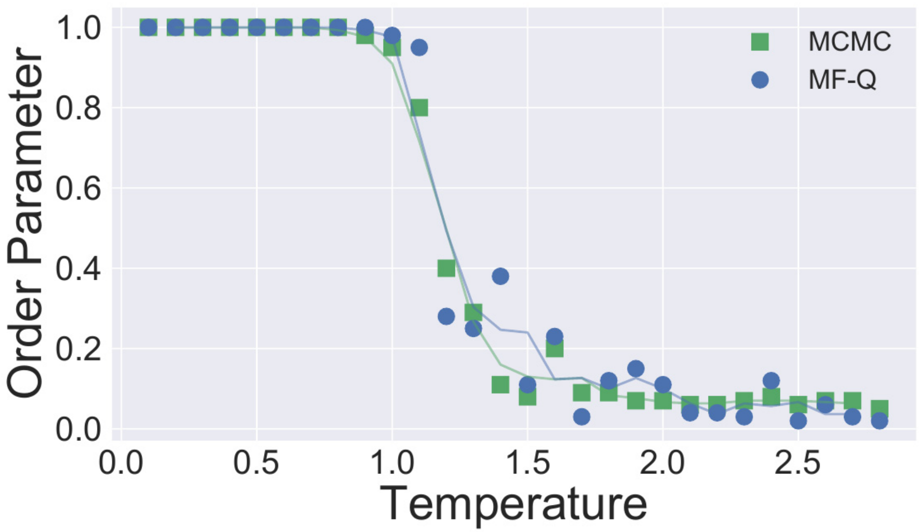
- Each spin is an agent to decide up or down (action)
- Measure: order parameter

$$\xi = \frac{|N_{\uparrow} - N_{\downarrow}|}{N}$$

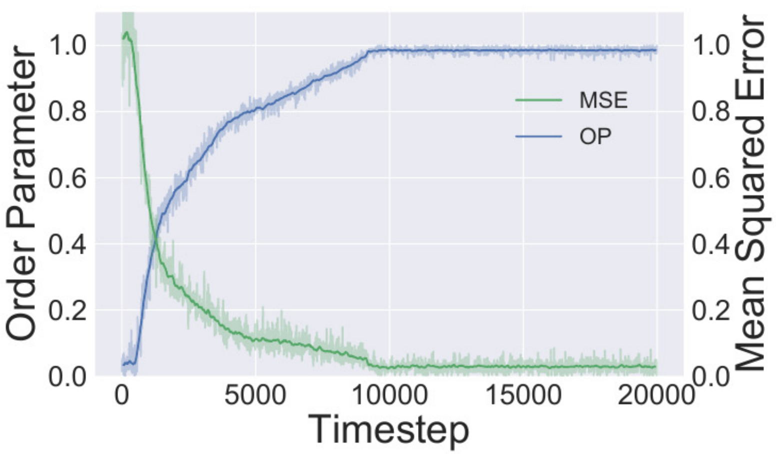
  - The closer OP is to 1, the more orderly the system is.



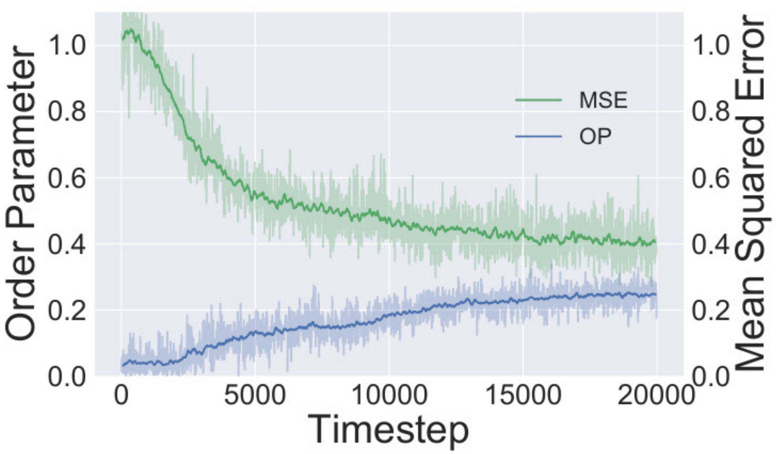
Experiment Performance IM



- Ground truth: MCMC simulation
- Goal: MF-Q learns with the similar behavior as MCMC, which we observed

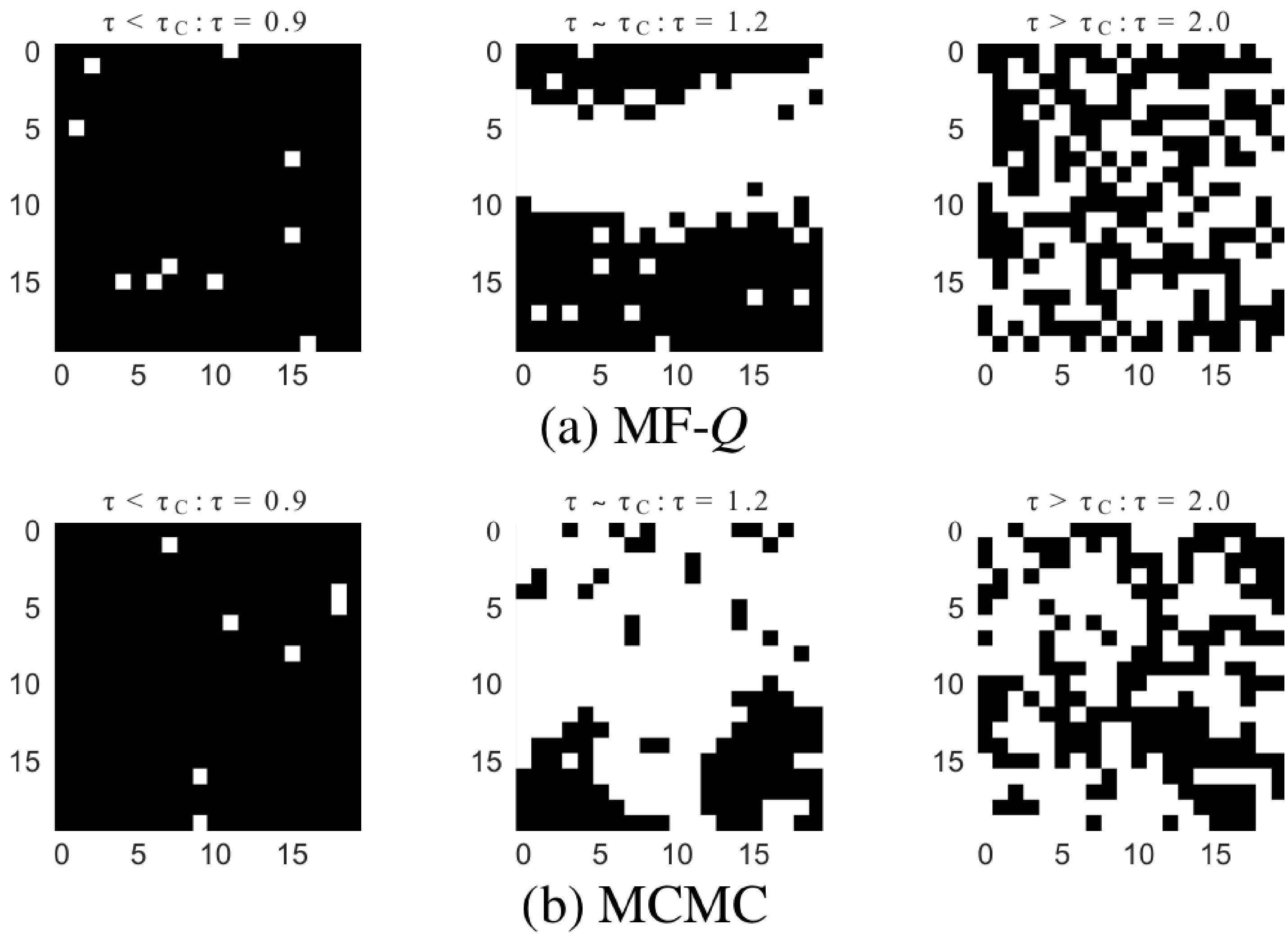


(a) $\tau = 0.8$



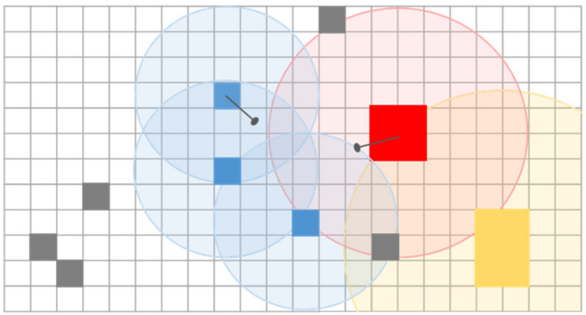
(b) $\tau = 1.2$

Experiment Performance IM

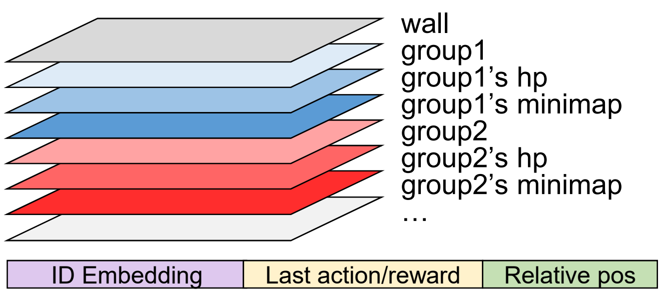


Experiment: Battle

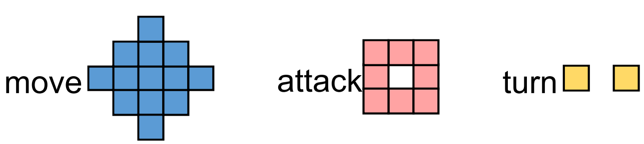
Grid World

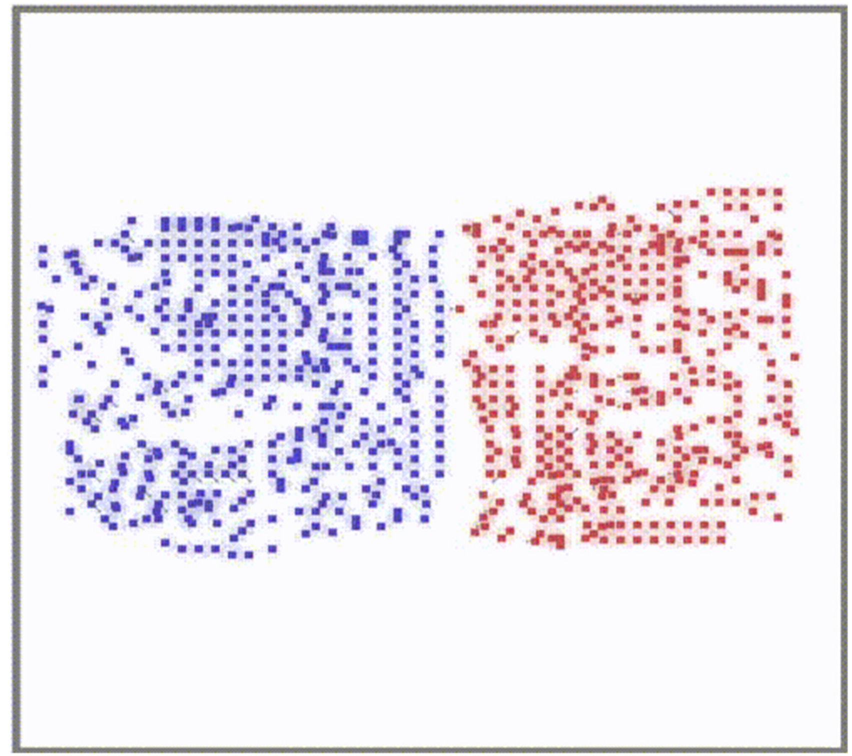


Observation Space



Action Space

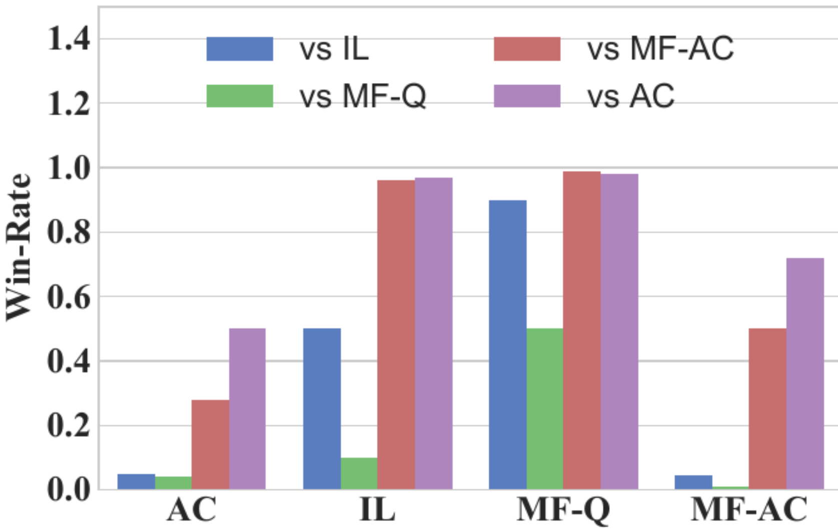




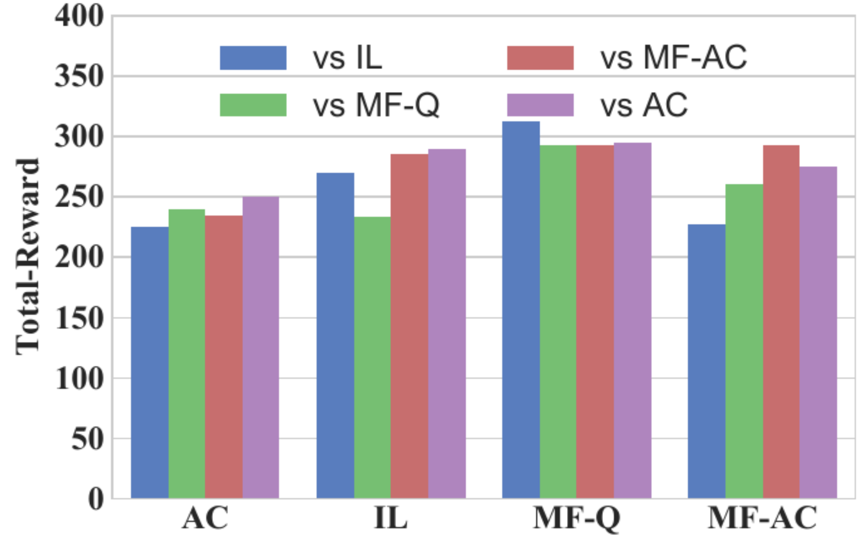
Supported by MAgent by Geek.AI

Lianmin Zheng, Weinan Zhang et al. "MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence." *NIPS 2017.*

Experiment Performance Battle



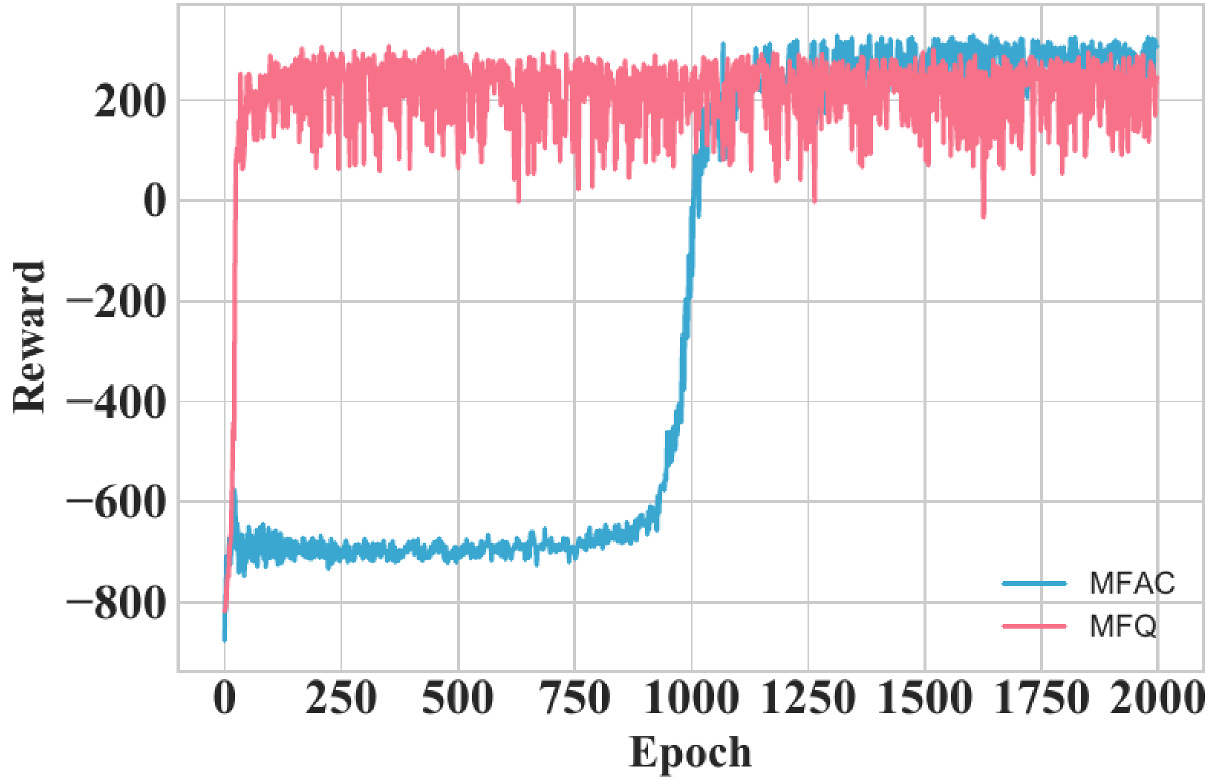
(a) Average wining rate.



(b) Average total reward.

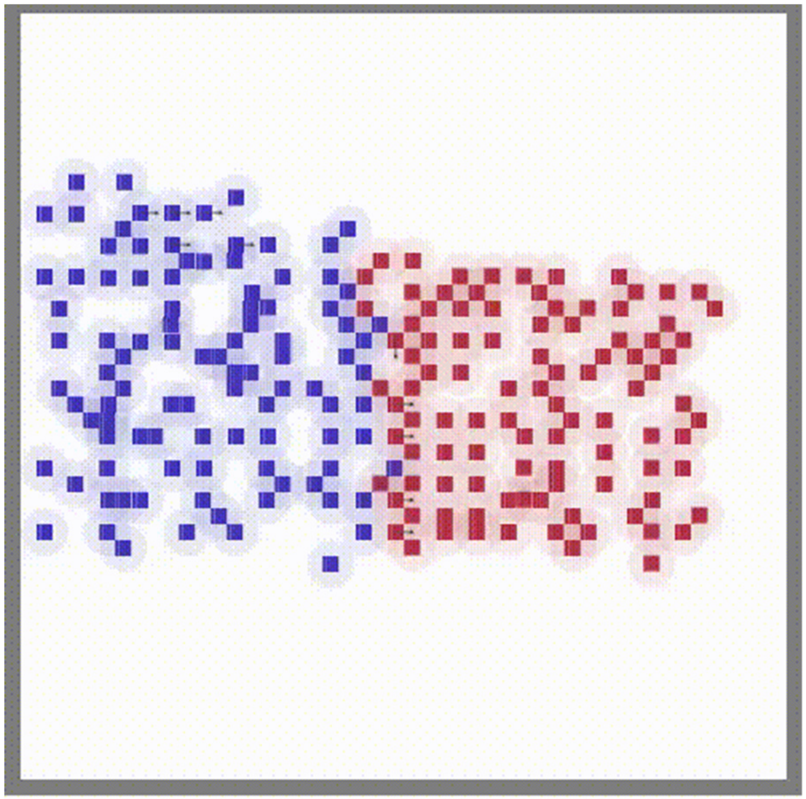
- For 64 vs 64 battle, MF-Q works the best among all compared models
- MF-AC may not work that well particularly when the agent number is large

Experiment Performance Battle



- MF-Q has a fast convergence property
- MF-AC has a phase changing point

Case Study



- Blue: MF-Q
- Red: IL

- MF-Q presents a go-around-and-besiege strategy

- MF-Q agents are more consistent with neighbors

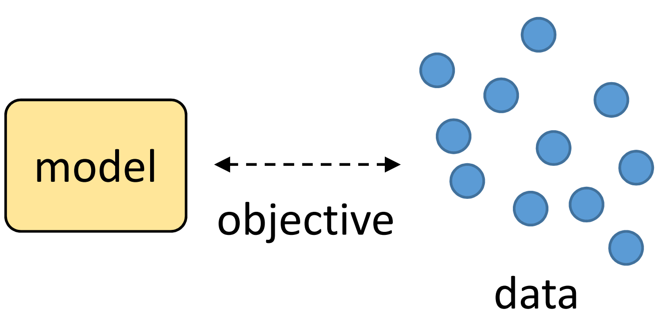
Summary of MARL

- Main difficulties for many-agent RL
  - Computational complexity
  - Complicated agent interactions
  - Highly dynamic neighborhood
- Possible solutions
  - Mean field approximation
  - MAgent platform

Summary from Machine Learning Perspective

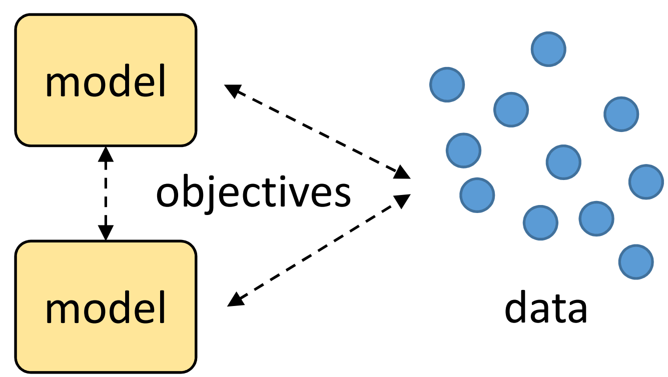
- Traditional machine learning is to build
  - a loss function
  - a likelihood estimation
  - an expectation of value

  from a machine and the training data and to optimize the objective



- Two-agent machine learning is to build
  - a loss function
  - a likelihood estimation
  - an expectation of value

  from the two machines and the training data and to optimize the objective



Summary Machine Learning Paradigm Extension