Multi-agent Reinforcement Learning (MARL)

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

November 14, 2019

Overview

References

- Egorov, Maxim. "Multi-agent deep reinforcement learning." CS231n: Convolutional Neural Networks for Visual Recognition (2016). (paper)
- Scaling Multi-Agent Reinforcement Learning by Eric Liang and Richard Liaw. OpenAI (blog post)
- Decision Making under Uncertainty by Mykel J. Kochenderfer
- Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in Neural Information Processing Systems. 2017.
- Multiagent Reinforcement Learning presentation by Marc Lanctot RLSS @ Lille, July 11th 2019 (pdf)
- Multiagent Learning Foundations and Recent Trends by Stefano Albrecht and Peter Stone Tutorial at IJCAI 2017 conference (presentation)

Table of Contents

Recap: Markov Models

	No Agents	Single Agent	Multiple Agents
State Known			
State Observed Indirectly			

Recap: Markov Models

	No Agents	Single Agent	Multiple Agents
State Known	Markov Chain	Markov Decision Process (MDP)	Markov Game (a.k.a. Stochastic Game)
State Observed Indirectly	Hidden Markov Model (HMM)	Partially-Observable Markov Decision Process (POMDP)	Partially-Observable Stochastic Game (POSG)

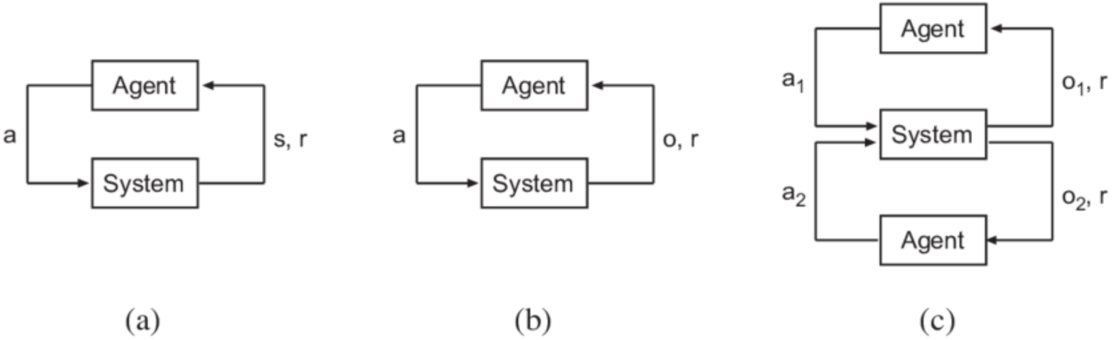
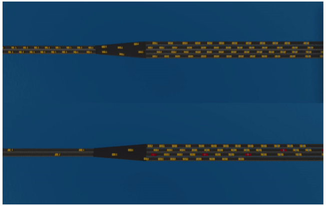


Figure: (a) Markov decision process (MDP) (b) Partially observable Markov decision process (POMDP) (c) Decentralized partially observable Markov decision process with two agents (Dec-POMDP)

Table of Contents

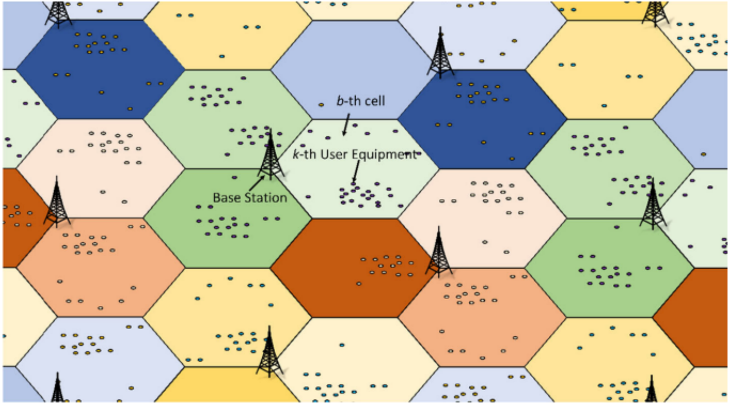
Multi-agent Applications: Traffic congestion reduction

- By intelligently controlling the speed of a few autonomous vehicles we can drastically increase the traffic flow
- Multi-agent can be a requirement here, since in mixed-autonomy settings, it is unrealistic to model traffic lights and vehicles as a single agent, which would involve the synchronization of observations and actions across all agents in a wide area.
- Flow Project website
- Flow simulation, without AVs and then with AV agents (red vehicles)



Multi-agent Applications: Antenna tilt control

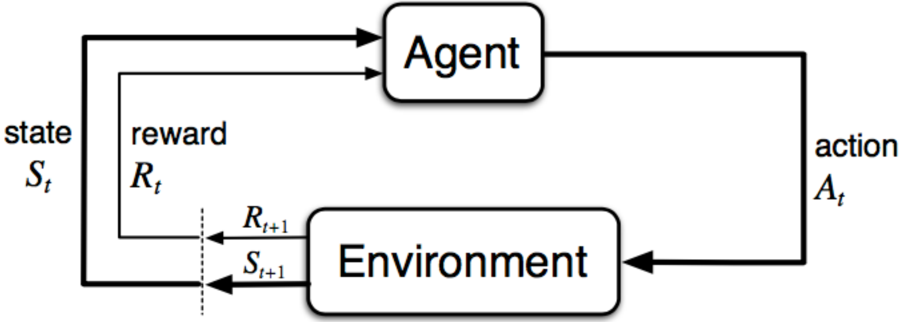
- The joint configuration of cellular base stations can be optimized according to the distribution of usage and topology of the local environment.
- Each base station can be modeled as one of multiple agents covering a city.



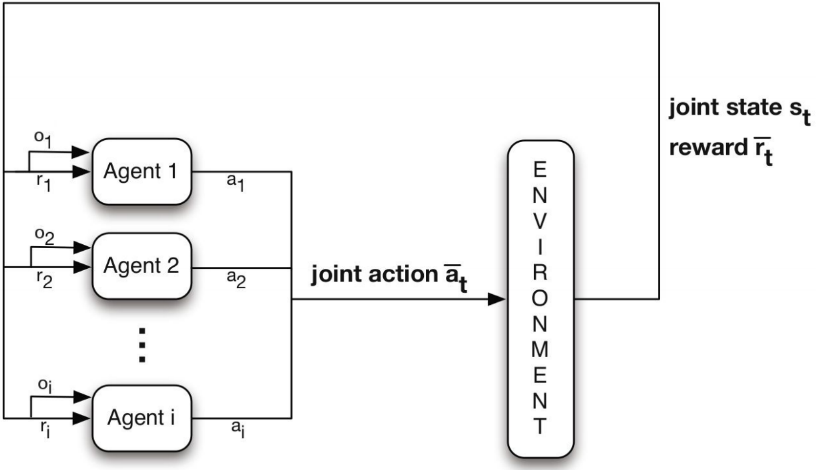
Multi-agent Applications: OpenAI Five

- Dota 2 AI agents are trained to coordinate with each other to compete against humans.
- Each of the five AI players is implemented as a separate neural network policy and trained together with large-scale PPO.

Markov Decision Process (MDP)



Multi-agent Reinforcement Learning (MARL)



Source: Nowe, Vrancx & De Hauwere 2012

Research in MARL

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Approximate Solution Methods	Approximate Solution Methods
Small Problems	Tabular Solution Methods	Tabular Solution Methods

Axes of MARL I

Centralized:

- One brain / algorithm deployed across many agents

Axes of MARL I

Centralized:

- One brain / algorithm deployed across many agents

Decentralized:

- All agents learn individually
- Communication limitations defined by environment

Axes of MARL II

Prescriptive:

- Suggests how agents should behave

Axes of MARL II

Prescriptive:

- Suggests how agents should behave

Descriptive:

- Forecast how agent will behave

Axes of MARL III

- **Cooperative:** Agents cooperate to achieve a goal
    - Shared team reward

Axes of MARL III

- **Cooperative:** Agents cooperate to achieve a goal
  - Shared team reward
- **Competitive:** Agents compete against each other
  - Zero-sum games
  - Individual opposing rewards

Axes of MARL III

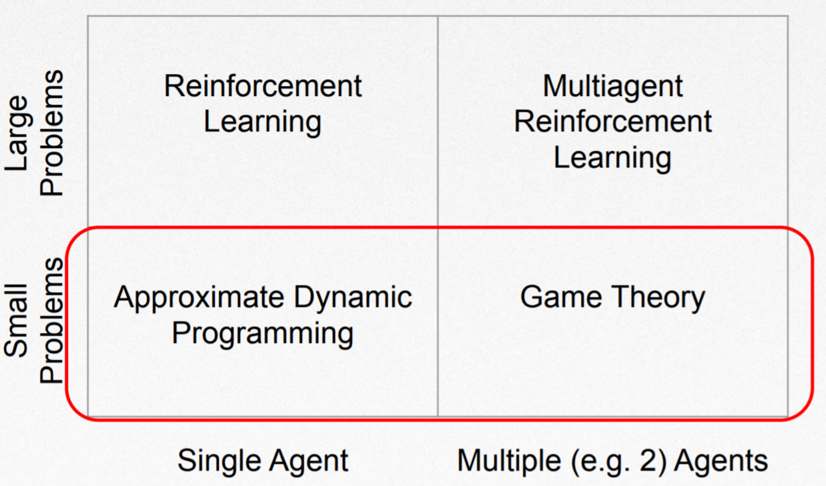
- **Cooperative:** Agents cooperate to achieve a goal
  - Shared team reward
- **Competitive:** Agents compete against each other
  - Zero-sum games
  - Individual opposing rewards
- **Neither:** Agents maximize their utility which may require cooperating and/or competing
  - General-sum games

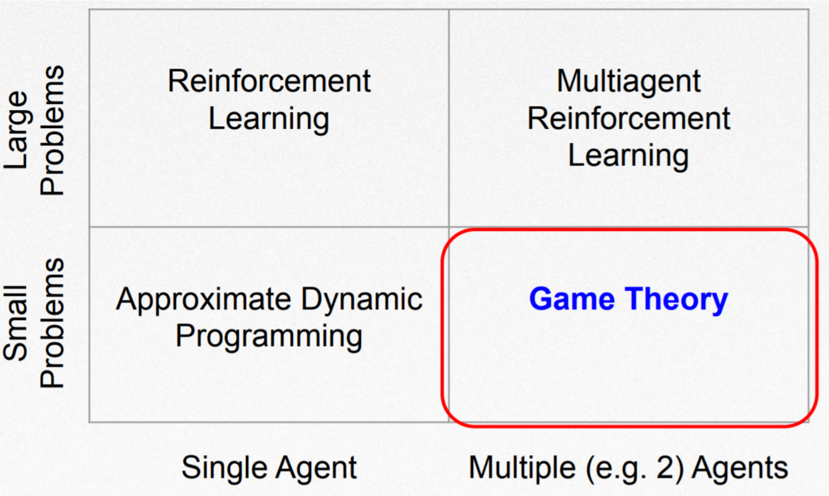
Axes of MARL VI

Numbers of agents

- One (single-agent)
- Two (very common)
- Finite
- Infinite

Foundations of (MA)RL




Large Problems
Small Problems
Reinforcement Learning
Multiagent Reinforcement Learning
Approximate Dynamic Programming
Game Theory
Single Agent
Multiple (e.g. 2) Agents

Multiagent Models

- Normal-form game
- Repeated game
- Stochastic game

Normal-Form "One-Shot" Game

Normal-form game consists of:

- Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$

Normal-Form "One-Shot" Game

Normal-form game consists of:

- Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$
- Each agent $i \in N$ has a set of actions $A_i \in \{a_1, a_2, \cdots\}$

Normal-Form "One-Shot" Game

Normal-form game consists of:

- Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$
- Each agent $i \in N$ has a set of actions $A_i \in \{a_1, a_2, \cdots\}$
- Set of **joint** actions $A = a_1 \times a_2 \times \cdots \times a_n$

Normal-Form "One-Shot" Game

Normal-form game consists of:

- Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$
- Each agent $i \in N$ has a set of actions $A_i \in \{a_1, a_2, \cdots\}$
- Set of **joint** actions $A = a_1 \times a_2 \times \cdots \times a_n$
- Rewards function $r_i : A \rightarrow \mathbb{R}$, where $A = A_1 \times \cdots \times A_n$

Normal-Form "One-Shot" Game

Normal-form game consists of:

- Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$
- Each agent $i \in N$ has a set of actions $A_i \in \{a_1, a_2, \cdots\}$
- Set of **joint** actions $A = a_1 \times a_2 \times \cdots \times a_n$
- Rewards function $r_i : A \to \mathbb{R}$, where $A = A_1 \times \cdots \times A_n$

Each agent i selects policy $\pi_i : A_i \to [0, 1]$, takes action $a_i \in A_i$ with probability $\pi_i(a_i)$, and receives reward $r_i(a_1, \cdots, a_n)$.

Normal-Form "One-Shot" Game

Normal-form game consists of:

- Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$
- Each agent $i \in N$ has a set of actions $A_i \in \{a_1, a_2, \cdots\}$
- Set of **joint** actions $A = a_1 \times a_2 \times \cdots \times a_n$
- Rewards function $r_i : A \to \mathbb{R}$, where $A = A_1 \times \cdots \times A_n$

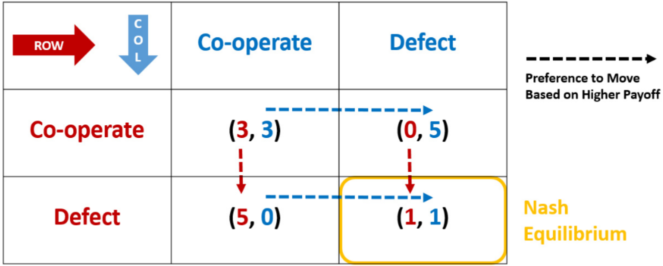
Each agent i selects policy $\pi_i : A_i \to [0, 1]$, takes action $a_i \in A_i$ with probability $\pi_i(a_i)$, and receives reward $r_i(a_1, \cdots, a_n)$. Given policy profile $(\pi_1, \cdots, \pi_n)$, expected reward to i is

$$r(\pi_1, \cdots, \pi_n) = \sum_{a \in A} \pi_1(a_1) * \cdots \pi_n(a_n) * r_i(a)$$

Agents want to maximise their expected reward.

Normal-Form Game: Prisoner's Dilemma

- Two prisoners questioned in isolated cells
- Each prisoner can Cooperate or Defect
- Rewards (row = agent 1, column = agent 2)



Normal-Form Game: Chicken

- Two opposite drivers on the same lane
- Each driver can Stay on lane or Leave lane

	S	L
S	0,0	7,2
L	2,7	6,6

Normal-Form Game: Rock-Paper-Scissors

- Two players, three actions
- Rock beats Scissors beats Paper beats Rock

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Repeated Game

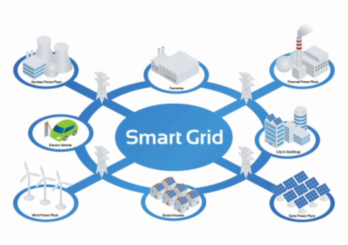
- Normal-form game is single interaction. No experience!
- Experience comes from repeated interactions

More Examples of MARL

Negotiation

Wireless networks

Smart grid

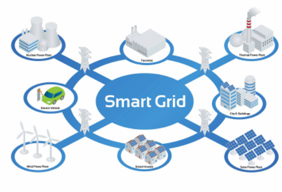


More Examples of MARL

Negotiation

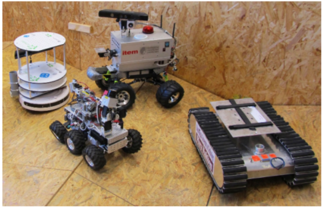
Wireless networks

Smart grid



User interfaces

Multi-robot rescue



Benefits of Multi-agent Learning Systems

- **Sharing experience**
  via communication, teaching, imitation

Benefits of Multi-agent Learning Systems

- **Sharing experience**
  via communication, teaching, imitation
- **Parallel computation**
  due to decentralized task structure

Benefits of Multi-agent Learning Systems

- **Sharing experience**
  via communication, teaching, imitation
- **Parallel computation**
  due to decentralized task structure
- **Robustness**
  redundancy, having multiple agents to accomplish a task

Challenges

Click!

Challenges in Multi-agent Learning Systems

- **Curse of dimensionality**
  Exponential growth in computational complexity from increase in state and action dimensions. Also a challenge for single-agent problems.

Challenges in Multi-agent Learning Systems

- **Curse of dimensionality**
  Exponential growth in computational complexity from increase in state and action dimensions. Also a challenge for single-agent problems.
- **Specifying a good (learning) objective**
  Agent returns are correlated and cannot be maximized independently.

Challenges in Multi-agent Learning Systems

- **Curse of dimensionality**
  Exponential growth in computational complexity from increase in state and action dimensions. Also a challenge for single-agent problems.
- **Specifying a good (learning) objective**
  Agent returns are correlated and cannot be maximized independently.
- **The system in which to learn is a moving target**
  As some agents learn, the system which contains these agents changes, and so may the best policy. Also called a system with non-stationary or time-dependent dynamics.

Challenges in Multi-agent Learning Systems

- **Curse of dimensionality**
  Exponential growth in computational complexity from increase in state and action dimensions. Also a challenge for single-agent problems.
- **Specifying a good (learning) objective**
  Agent returns are correlated and cannot be maximized independently.
- **The system in which to learn is a moving target**
  As some agents learn, the system which contains these agents changes, and so may the best policy. Also called a system with non-stationary or time-dependent dynamics.
- **Need for coordination**
  Agent actions affect other agents and could confuse other agents (or herself) if not careful. Also called destabilizing training.

Challenges: Non-stationarity of Environment

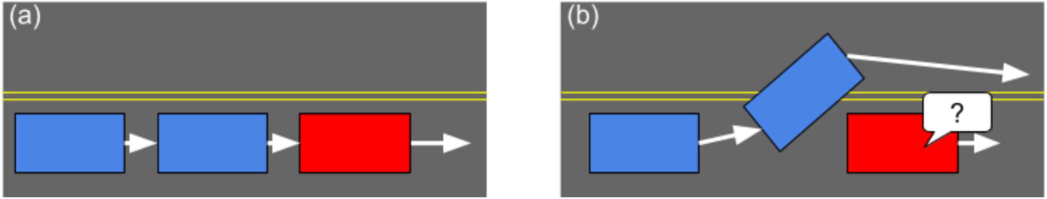


***Figure 2**: Non–stationarity of environment: Initially (a), the red agent learns to regulate the speed of the traffic by slowing down. However, over time the blue agents learn to bypass the red agent (b), rendering the previous experiences of the red agent invalid.*

Challenges: High Variance of Estimates

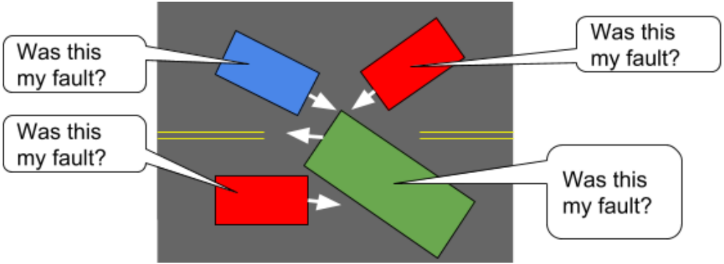


***Figure 4: High variance of advantage estimates**: In this traffic gridlock situation, it is unclear which agents' actions contributed most to the problem -- and when the gridlock is resolved, from any global reward it will be unclear which agents get credit.*

Summary of Challenges

- In single agent RL, agents need only to adapt their behaviour in accordance with their own actions and how they change the environment. In MARL agents also need to adapt to other agents' learning and actions. The effect is that agents can execute the same action on the same state and receive different rewards.

Summary of Challenges

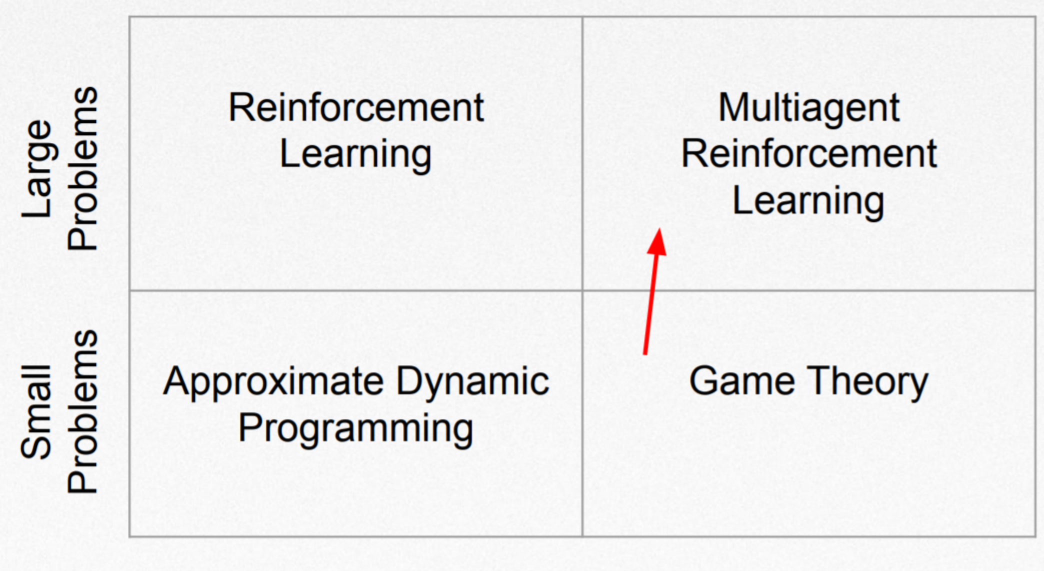
- In single agent RL, agents need only to adapt their behaviour in accordance with their own actions and how they change the environment. In MARL agents also need to adapt to other agents' learning and actions. The effect is that agents can execute the same action on the same state and receive different rewards.
- MARL agents do not always have a full view of the environment and even if they have, they normally cannot predict the actions of other agents and the changes in the environment

Summary of Challenges

- In single agent RL, agents need only to adapt their behaviour in accordance with their own actions and how they change the environment. In MARL agents also need to adapt to other agents' learning and actions. The effect is that agents can execute the same action on the same state and receive different rewards.
- MARL agents do not always have a full view of the environment and even if they have, they normally cannot predict the actions of other agents and the changes in the environment
- The credit assignment problem - the difficulty of deciding which agent is responsible for successes or failures. How to split the reward signal among the agents and the trade-off between the use of local and global rewards to achieve fast learning or to guarantee to converge to a global optimal policy.

Table of Contents

Foundations of (MA)RL



First MARL Algorithm: Minimax-Q (Littman '94)

Q-values are over joint actions: $Q(s, a, o)$

- s = state
- a = your action
- o = action of the opponent

First MARL Algorithm: Minimax-Q (Littman '94)

Q-values are over joint actions: $Q(s, a, o)$

- s = state
- a = your action
- o = action of the opponent

Instead of playing action with highest $Q(s, a, o)$, play **MaxMin**

$$Q(s, a, o) = (1 - \alpha)Q(s, a, o) + \alpha(r + \gamma V(s'))$$

$$V(s) = \max_{\pi_s} \min_{o} \sum_{a} Q(s, a, o)\pi_s(a)$$

MARL Formulation

- The agents choose actions according to their policies.

MARL Formulation

- The agents choose actions according to their policies.
- For agent j, the corresponding policy is defined as $\pi^j : S \to \Omega(A^j)$, where $\Omega(A^j)$ is the collection of probability distributions over agent j's action space A^j.

MARL Formulation

- The agents choose actions according to their policies.
- For agent j, the corresponding policy is defined as $\pi^j : S \to \Omega(A^j)$, where $\Omega(A^j)$ is the collection of probability distributions over agent j's action space A^j.
- Let $\boldsymbol{\pi} = [\pi^1, \cdots, \pi^N]$ - is the joint policy of all agents, then

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

MARL Formulation

- The agents choose actions according to their policies.
- For agent j, the corresponding policy is defined as $\pi^j : S \rightarrow \Omega(A^j)$, where $\Omega(A^j)$ is the collection of probability distributions over agent j's action space A^j.
- Let $\boldsymbol{\pi} = [\pi^1, \cdots, \pi^N]$ - is the joint policy of all agents, then

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

- Q-function such that the Q-function $Q^j_{\boldsymbol{\pi}} : S \times A^1 \times \cdots \times A^N \rightarrow \mathbb{R}$ of agent j under the joint policy $\boldsymbol{\pi}$:

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

Nash Q-learning

- In MARL, the objective of each agent is to learn an optimal policy to maximize its value function

Nash Q-learning

- In MARL, the objective of each agent is to learn an optimal policy to maximize its value function
- Optimizing the v_π^j for agent j depends on the joint policy $\boldsymbol{\pi}$ of all agents

Nash Q-learning

- In MARL, the objective of each agent is to learn an optimal policy to maximize its value function
- Optimizing the $v^j_{\boldsymbol{\pi}}$ for agent j depends on the joint policy $\boldsymbol{\pi}$ of all agents
- A Nash equilibrium is a joint policy π such that no player has incentive to deviate unilaterally. It is represented by a particular joint policy

$$\boldsymbol{\pi_*} = [\pi^1_*, \cdots, \pi^N_*]$$

such that for all $s \in S, j \in \{1, \cdots, N\}$ it satisfies:

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

Here $\boldsymbol{\pi}^{\boldsymbol{-j}}_*$ is the joint policy of all agents except j as

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

Nash Q-learning

- In a Nash equilibrium, each agent acts with the best response π_*^j to others, provided that all other agents follow the policy $\boldsymbol{\pi}_*^{-j}$

Nash Q-learning

- In a Nash equilibrium, each agent acts with the best response π_*^j to others, provided that all other agents follow the policy $\boldsymbol{\pi}_*^{-j}$
- For a N-agent stochastic game, there is at least one Nash equilibrium with stationary policies, assuming players are rational

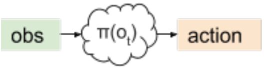
Nash Q-learning

- In a Nash equilibrium, each agent acts with the best response π_*^j to others, provided that all other agents follow the policy $\boldsymbol{\pi}_*^{-j}$
- For a N-agent stochastic game, there is at least one Nash equilibrium with stationary policies, assuming players are rational
- Given Nash policy $\boldsymbol{\pi}_*$, the Nash value function

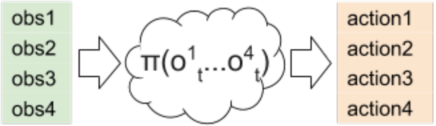
$$\boldsymbol{v}^{\text{Nash}} = [v^1_{\boldsymbol{\pi}_*}(s), \cdots, v^N_{\boldsymbol{\pi}_*}(s)]$$
$$\boldsymbol{Q}(s, \boldsymbol{a})^{\text{Nash}} = \mathbb{E}_{s' \sim p}[\boldsymbol{r}(s, \boldsymbol{a}) + \gamma \boldsymbol{v}^{\text{Nash}}(s')]$$

where $\boldsymbol{r}(s, \boldsymbol{a}) = [r^1(s, \boldsymbol{a}), \cdots, r^N(s, \boldsymbol{a})]$

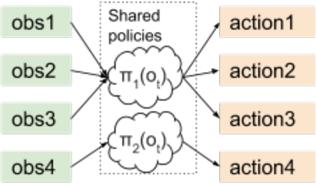
MARL Policies



(a) Single-agent



(b) Multiple logical entities, single "super-agent"



(c) Multi-agent

MARL Policies

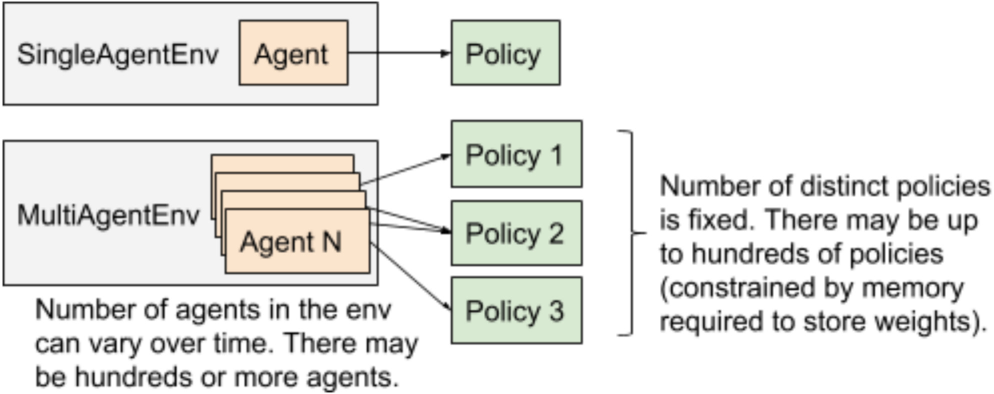
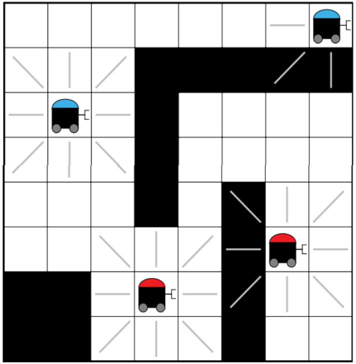


Table of Contents

Multi-agent Deep Q-Network (MADQN) Example: Pursuit Evasion

- n pursuit-evasion – a set of agents (the pursuers) are attempting to chase another set of agents (the evaders)
- The agents in the problem are self-interested (or heterogeneous), i.e. they have different objectives
- The two pursuers are attempting to catch the two evaders



Multi-agent Deep Q-Network (MADQN): Problem representation

Challenge: defining the problem in such a way that an arbitrary number of agents can be represented without changing the architecture of the deep Q-Network.

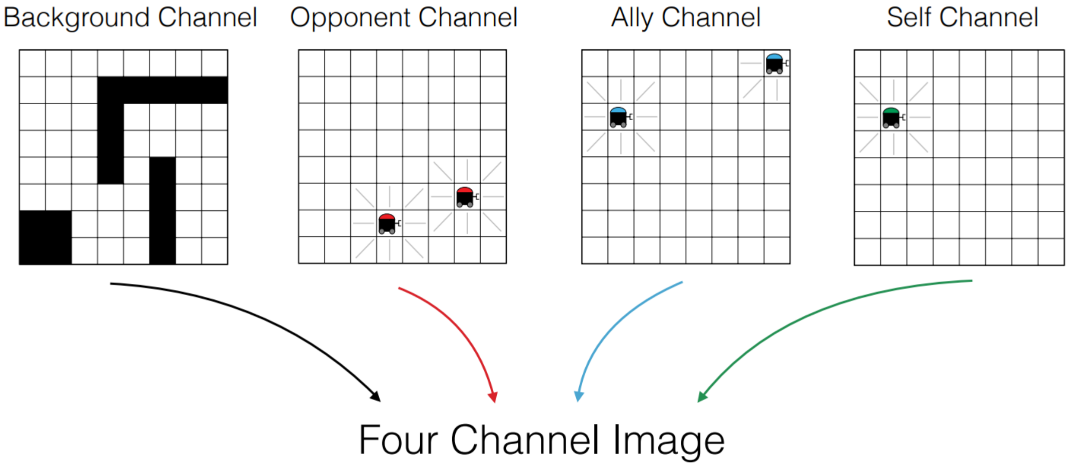
Multi-agent Deep Q-Network (MADQN): Problem representation

Challenge: defining the problem in such a way that an arbitrary number of agents can be represented without changing the architecture of the deep Q-Network.

Solution (under some assumptions):

- The image tensor is of size $4\times$ W $\times$ H, where W and H are the height and width of our two dimensional domain and four is the number of channels in the image.
- Channels:
  - **Background Channel**: contains information about any obstacles in the environment
  - **Opponent Channel:** contains information about all the opponents
  - **Ally Channel:** contains information about all the allies
  - **Self Channel:** contains information about the agent making the decision

Multi-agent Deep Q-Network (MADQN): Four Channel Image



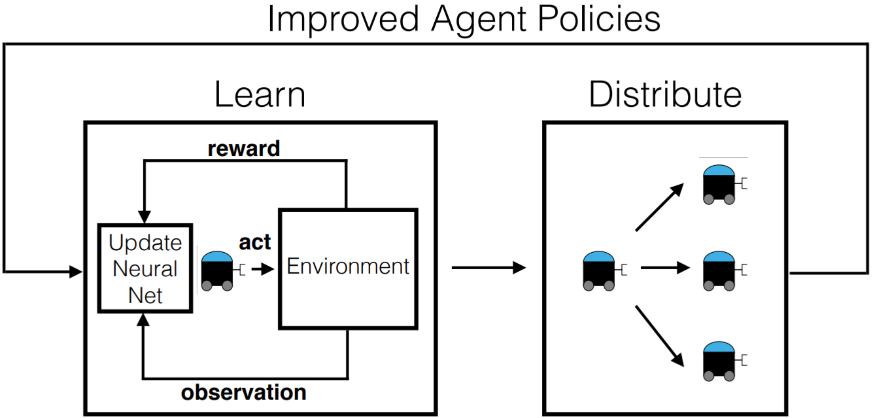
MADQN: Multi-agent Centralized Training

- Train one agent at a time, and fix policies of all the other agents

MADQN: Multi-agent Centralized Training

- Train one agent at a time, and fix policies of all the other agents
- After a number of iterations distribute the policy learned by the training agent to all the other agents of its type

MADQN: Multi-agent Centralized Training



MADQN: Dealing with agent ambiguity

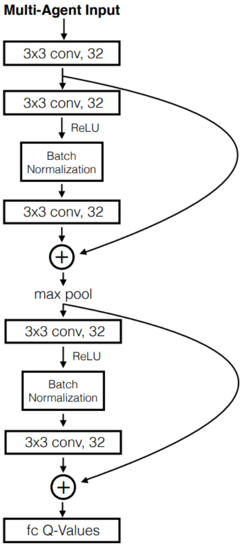
- **Challenge:** When two ally agents are occupying the same position in the environment, the image-like state representation for each agent will be identical, so their policies will be exactly the same.

MADQN: Dealing with agent ambiguity

- **Challenge:** When two ally agents are occupying the same position in the environment, the image-like state representation for each agent will be identical, so their policies will be exactly the same.
- **Solution:** To break this symmetry – use a stochastic policy for agents. The actions taken by the agent are drawn from a distribution derived by taking a softmax over the Q-values of the neural network. This allows allies to take different actions if they occupy the same state and break the ambiguity.

MADQN Architecture: Residual Network Type

MADQN architecture – a Residual Network type architecture is used to improve gradient flow throughout the network



MADQN Architecture