Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

Ozsel Kilinc Giovanni Montana

WMG, Data Science Group
University of Warwick

WCPM Seminar, 2019

Outline

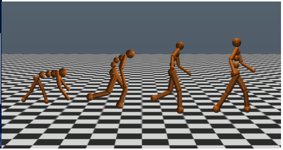
1. Reinforcement Learning [1]
   - RL in a Nutshell
   - RL Basics
   - Deep Reinforcement Learning
2. Multi-agent Reinforcement Learning
   - MARL Basics
   - Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

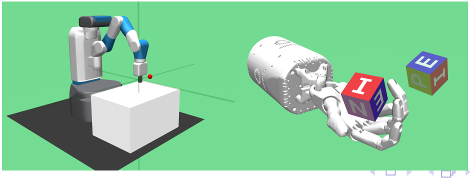
Outline

1. Reinforcement Learning [1]
   - RL in a Nutshell
   - RL Basics
   - Deep Reinforcement Learning
2. Multi-agent Reinforcement Learning
   - MARL Basics
   - Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

Reinforcement Learning

- Defines very general framework for sequential decision-making
  - Play Atari games, Go, StarCraft
  - Make a humanoid walk
  - Robotics
- Learning by trial-and-error
- Improves with experience



Recent Success

- 2013, DQN in Atari
  - Learning to play many classic Atari games with human performance
- 2016, AlphaGo
  - Learning to play Go, and win against 18-time world champion by 4-1
  - Initially trained on thousands of human amateur and professional games to learn how to play Go
- 2017, AlphaGo Zero
  - World's best Go Player
  - No initial training, learns to play simply by playing games against itself, starting from completely random play
- 2017-2108, Deep RL in Robotics
  - Learning of locomotion behaviours in rich environments
  - Learning dexterity

RL vs. Supervised Learning

- Data
  - Non-i.i.d, sequential data
  - Depends on agent's actions
- Supervision
  - No ground-truth labels, only a reward signal
  - Mostly delayed, sometimes very sparse

RL Challenges

- Sample inefficiency: e.g. 200 years of real-time play experience
- Reproducibility: More sensitive to hyper-parameters and random seeds than supervised learning
- Long term credit assignment: Feedback is not immediate. Which series of actions are actually responsible for the high reward?
- High variance

Outline

1. Reinforcement Learning [1]
   - RL in a Nutshell
   - RL Basics
   - Deep Reinforcement Learning
2. Multi-agent Reinforcement Learning
   - MARL Basics
   - Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

The RL Problem

- **Environment:** Markov Decision Process (MDP)
- **Agent:** Decision maker

Definition (MDP)

A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{T}$ is a state transition function
- $\mathcal{R}$ is a reward function
- γ is a discount factor

The RL Problem

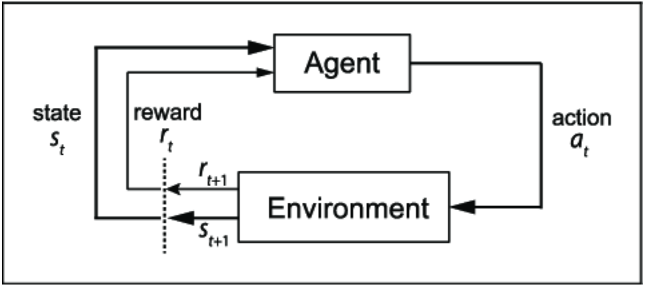
Definition (Markov Property)

A state s^t is *Markov* iff $\mathbb{P}[s^{t+1}|s^t] = \mathbb{P}[s^{t+1}|s^1, \ldots, s^t]$

- Current state captures all relevant information from the history
- If s^t is known, $s^1, \ldots, s^{t-1}$ may be thrown away

The RL Problem

1. **Environment** emits state $s^t \in \mathcal{S}$
2. **Agent** executes action $a^t = \boldsymbol{\mu}(s^t) : \mathcal{S} \to \mathcal{A}$
3. **Environment** emits scalar reward $r^t = \mathcal{R}(s^t, a^t)$
4. **Environment** emits next state $s^{t+1} = \mathcal{T}(s^t, a^t)$



The RL Problem

- *Reward* r^t
  - Scalar feedback signal provided by the **environment**, e.g. AV
    - $+$ if the AV reaches the destination
    - $-$ for the time spent on the road
    - $-$ for accident
  - Indicates how well **agent** is doing at step t
  - Actions may have long term consequences, rewards may be delayed
  - Sacrificing immediate reward r^t may bring more in the future
    - e.g. Refuelling may help go further

The RL Problem

- *Return* R^t
  - The total discounted rewards from time-step t
  - $r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^T r^{t+T}$
  - The discount factor $\gamma \in [0, 1]$
    - $\gamma \to 0$: "myopic" evaluation
    - $\gamma \to 1$: "far-sighted" evaluation
  - The **agent**'s goal is to select actions to maximise $\mathbb{E}[R^t]$

Components of an RL Agent

- Policy: **Agent**'s behaviour
  - A mapping from state domain to action domain $\mathcal{S} \to \mathcal{A}$
  - Deterministic policy $\boldsymbol{\mu}$, i.e. $a^t = \boldsymbol{\mu}(s^t)$
  - Stochastic policy $\boldsymbol{\pi}$, i.e. $\boldsymbol{\pi}(a|s^t) = \mathbb{P}[a^t = a|s^t]$

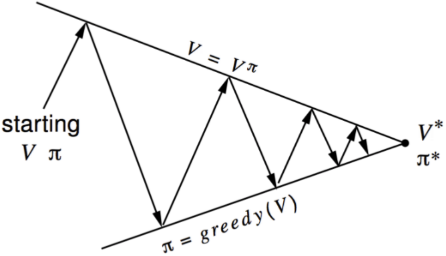
Components of an RL Agent

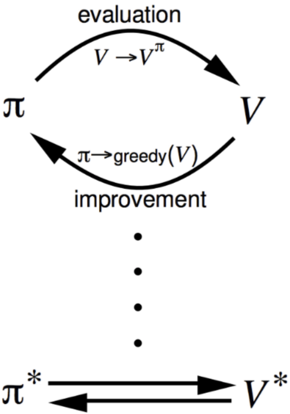
- Value function: Prediction of the Return
  - To evaluate the goodness/badness of states and/or actions
  - To select between the actions
  - The state-value function $V^{\pi}(s) = \mathbb{E}_{\pi}[R^t | s^t = s]$
  - The action-value function $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[R^t | s^t = s, a^t = a]$

Components of an RL Agent

- Model: **Agent**'s representation of the **environment**
  - A model to predict what the **environment** will do next
  - Estimations of the state transition function $\mathcal{T}$ and the reward function $\mathcal{R}$
- Learning vs. Planning
  - Learning: Model is unknown, **agent** interacts with the **environment**
  - Planning: Model is known, **agent** performs computations with its model

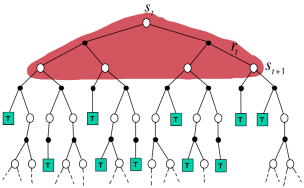
Learning the Optimal Policy



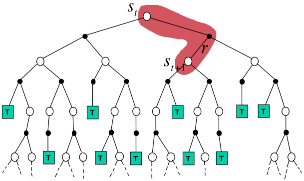


- Policy evaluation: Estimate V^π or Q^π
- Policy improvement: Generate $\pi' \geq \pi$

Policy Evaluation $V \rightarrow V^{\pi}$



$V(s^t) \leftarrow \mathbb{E}[r^t + \gamma V(s^{t+1}) \,|\, s^t]$



$V(s^t) \overset{\alpha}{\leftarrow} r^t + \gamma V(s^{t+1})$

* shallow backups
* uses estimated return

$r^t + \gamma V^{\pi}(s^{t+1})$

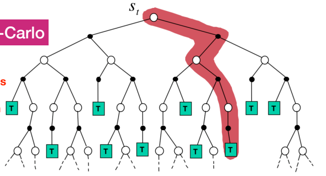
Dynamic programming

TD-learning

Bootstrapping

$V^{\pi}(s^t) = \mathbb{E}[R^t \,|\, s^t]$

$V(s^t) \overset{\alpha}{\leftarrow} R^t$



* deep backups
* uses actual return

$R^t = r^t + \gamma r^{t+1} + \dots$

Exhaustive search

Monte-Carlo

Sampling

* full backups
* model-based
* expectation

* sample backups
* model-free
* empirical mean

Temporal-Difference Learning

- Model-free: No knowledge of MDP
- Exploits Markov property
- Can learn online after every step
- Can learn from incomplete sequences, by bootstrapping

Policy Improvement $\pi' \geq \pi$

- Improve the policy by acting greedily w.r.t. V^{π}
- $\pi' = \text{greedy}(V^{\pi})$
- When using sample backups, exploration becomes important
- ϵ-greedy:
  - With probability $1 - \epsilon$ choose the greedy action
  - With probability ϵ choose an action at random

Q-Learning

- Policy evaluation: Apply TD to $Q^{\pi}(s, a)$
- Policy improvement: Use ϵ-greedy

1. $Q(s^t, a^t) \xleftarrow{\alpha} r^t + \max_{a'} \gamma Q(s^{t+1}, a')$
2. $Q \to Q^{\pi}$
3. $\pi' \leftarrow \epsilon\text{-greedy}(Q^{\pi})$

Theorem

Q-Learning control converges to the optimal action-value function, $Q(s, a) \to Q^*(s, a)$

Outline

1. Reinforcement Learning [1]
   - RL in a Nutshell
   - RL Basics
   - Deep Reinforcement Learning
2. Multi-agent Reinforcement Learning
   - MARL Basics
   - Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

Deep RL

- So far we considered lookup tables to represent $V^{\pi}(s)$ and $Q^{\pi}(s, a)$
  - An entry per s, or s, a pair
- Problem: We want to solve large MDPs e.g. Go: 10^{170} states
  - Too many states and/or actions to store in memory
  - Too slow to learn each value individually
- Solution: Using function approximation such as Neural Networks
  - $V^{\pi_\theta}(s, \omega)$ and $Q^{\pi_\theta}(s, a, \omega)$
  - Generalise from seen to unseen
  - Learn parameters ω inside the RL paradigm using SGD

Convergence with Approximation

	Lookup table	Linear	Non-linear
Monte-Carlo	✓	(✓)	✗
Q-Learning	✓	✗	✗

- No theoretical guarantee, but empirically it works well
- Tricks to help Q-Learning work with NNs
  - Using experience replay
  - Using fixed Q-targets

Experience Replay

- Store all transitions (s^t, a^t, r^t, s^{t+1}) experienced by the agent in a replay buffer $\mathcal{D}$
- Update parameters ω using a mini-batch of transitions (s, a, r, s') sampled from $\mathcal{D}$
- Without XP: Updating ω using data $\sim \boldsymbol{\pi}^k$
- With XP: Updating ω using data $\sim \{\boldsymbol{\pi}^0, \boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^k\}$
- Stabilises the learning

Fixed Q-Targets

- Goal: Update ω to minimise $(Q^{\pi}(s, a; \omega) - target)^2$
- Problem: The value of *target* also changes with each update
  - $target = r + \gamma \max_{a'} Q^{\pi}(s', a'; \omega)$
- Solution: Compute targets w.r.t. old, fixed parameters ω'
  - $target = r + \gamma \max_{a'} Q^{\pi}(s', a'; \omega')$
- Once in every U steps, update ω' with ω and then keep fixed until next update
- Stabilises the learning

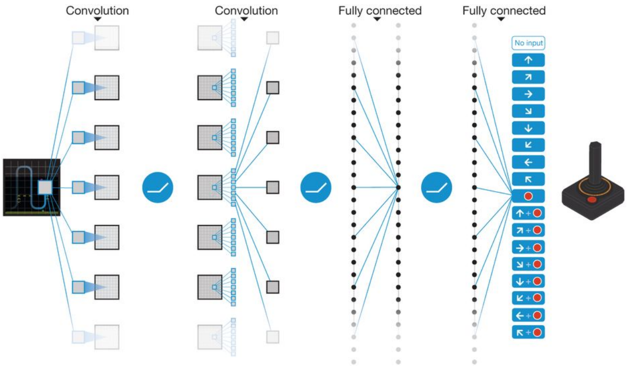
Deep Q-Networks (DQN) [2]

1. Take action a^t w.r.t. ϵ-greedy(Q^π)
2. Store transition (s^t, a^t, r^t, s^{t+1}) in replay memory $\mathcal{D}$
3. Sample a random mini-batch of transitions $(s, a, r, s') \sim \mathcal{D}$
4. Optimise MSE between the Q-Network and the target Q-Network using SGD

$$\begin{aligned} \mathcal{L}(\omega) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\left[(Q^\pi(s, a; \omega) - y)^2\right] \\ y = r + \gamma \max_{a'} Q^\pi(s', a'; \omega') \end{aligned} \tag{1}$$

Deep Q-Networks (DQN) in Atari [2]

- End-to-end learning from pixels to $Q^\pi(s, a)$
- State is stack of raw pixels from last 4 frames, $s^t \in \mathbb{R}^{4\times84\times84}$
- Action is one of 18 discrete joystick/button positions, $a^t \in \mathbb{R}^{18}$
- Reward is the change in the score



Actor-Critic RL

- So far we considered value-based RL
  - Policy evaluation: Learnt value function, e.g. $Q^{\pi}(s^t, a^t; \omega)$
  - Policy improvement: Implicit policy, e.g. $a^t = \arg\max_{a'} Q^{\pi}(s^t, a'; \omega)$
- What if we have continuous action space?
  - Greedy policy improvement becomes problematic
  - Requires a global maximisation at every step
- Actor-Critic RL
  - Policy evaluation: Learnt value function, e.g. $Q^{\pi}(s^t, a^t; \omega)$, i.e. *critic*
  - Policy improvement: Learnt policy $\pi(a|s^t; \theta)$, i.e. *actor*

Actor-Critic RL

- Goal: Update parameters θ to maximise $J(\theta) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\pi}}, a\sim\boldsymbol{\pi}_\theta}[R]$ by taking steps in the direction of $\nabla_\theta J(\theta)$
- Based on policy gradient theorem

Theorem (Policy Gradient Theorem [3])

For any differentiable policy $\boldsymbol{\pi}_\theta$, the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\pi}}, a\sim\boldsymbol{\pi}_\theta}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) Q^{\boldsymbol{\pi}}(s,a)\right]$$

Theorem (Deterministic Policy Gradient Theorem [4])

For any differentiable differentiable policy $\boldsymbol{\mu}_\theta$, the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^{\boldsymbol{\pi}}}\left[\nabla_\theta \boldsymbol{\mu}_\theta(s) \nabla_a Q^{\boldsymbol{\mu}}(s,a)|_{a=\boldsymbol{\mu}_\theta(s)}\right]$$

Deep Deterministic Policy Gradient (DDPG) [5]

- Adopts DPG
- Actor $\boldsymbol{\mu}$ and Critic $Q^{\boldsymbol{\mu}}$ are approximated with Deep NNs
- Similarly to DQN, employs experience replay and target network

1. Take action $a^t = \boldsymbol{\mu}(s^t; \theta) + \mathcal{N}^t$
2. Store transition (s^t, a^t, r^t, s^{t+1}) in replay memory $\mathcal{D}$
3. Sample a random mini-batch of transitions $(s, a, r, s') \sim \mathcal{D}$
4. Update the Critic by minimising the loss

$$\begin{aligned} \mathcal{L}(\omega) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\left[(Q^{\boldsymbol{\mu}}(s, a; \omega) - y)^2\right] \\ y = r + \gamma \max_{a'} Q^{\boldsymbol{\mu}}(s', a'; \omega') \end{aligned} \tag{2}$$

5. Update the Actor using the gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}}\left[\nabla_\theta \boldsymbol{\mu}(s; \theta) \nabla_a Q^{\boldsymbol{\mu}}(s, a; \omega)\big|_{a=\boldsymbol{\mu}(s;\theta)}\right] \tag{3}$$

Outline

1. Reinforcement Learning [1]
   - RL in a Nutshell
   - RL Basics
   - Deep Reinforcement Learning
2. Multi-agent Reinforcement Learning
   - MARL Basics
   - Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

The MARL Problem

- Partially observable Markov Games (POMGs)
  - Multi-agent extensions of MDPs of N agents

Definition (POMG [6])

A Partially Observable Markov Game is a tuple $G = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Q}, \mathcal{O}, \gamma, N \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a collection of sets of actions, $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$
- $\mathcal{T}$ is a state transition function
- $\mathcal{R}$ is a reward function
- $\mathcal{Q}$ is a collection of private observation functions $\mathcal{Q} = \{\mathcal{Q}_1, \ldots, \mathcal{Q}_N\}$
- $\mathcal{O}$ is a collection of private observations $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$
- γ is a discount factor
- N is the number of agents

The MARL Challenges

- Partial observability
  - Agents do not have full access to the true state s^t
  - Each agent receives a private partial observation o_i^t correlated with s^t
  - And chooses an action according to a policy conditioned on its own private observation, i.e. $a_i^t = \boldsymbol{\mu}(o_i^t; \theta_i)$

The MARL Challenges

- Non-stationarity
  - Environment moves into the next state s^{t+1} according to actions of all agents, i.e. $s^{t+1} = \mathcal{T}(s^t, a_1^t, \ldots, a_N^t)$
  - It is non-stationary from the viewpoint of any agent: when any $\boldsymbol{\mu}_i \neq \boldsymbol{\mu}'_i$ $\mathbb{P}(o_i^{t+1} | o_i^t, a_i^t, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N) \neq \mathbb{P}(o_i^{t+1} | o_i^t, a_i^t, \boldsymbol{\mu}'_1, \ldots, \boldsymbol{\mu}'_N)$
- Credit assignment
  - Which agent is responsible for the received reward?

The MARL Challenges

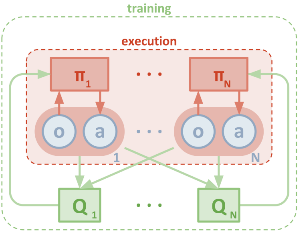
- Markov assumption is violated due to PO + NS
- Transitions in the experience replay become invalid due to NS
- High variance problem exacerbates due to CA
- Sample inefficiency exacerbates due to PO + NS + CA

Naive Solutions

- Ignore all the problems and train agents independently
  - Train in decentralised manner, i.e. $Q_i^{\boldsymbol{\mu}}(o_i, a_i)$
  - Execute in decentralised manner, i.e. $\boldsymbol{\mu}_i(o_i)$
  - Over-optimistic
- Use all available information and train agents as a single *Meta-agent*
  - Train in centralised manner, i.e. $Q_i^{\boldsymbol{\mu}}(o_1, a_1, \ldots, o_N, a_N)$
  - Execute in centralised manner, i.e. $\boldsymbol{\mu}_i(o_1, \ldots, o_N)$
  - Scalability: input size is multiplied by N for each one of N agents
  - In a realistic scenario where agents work remotely, $(N-1)N$ transmissions are required at each time-step

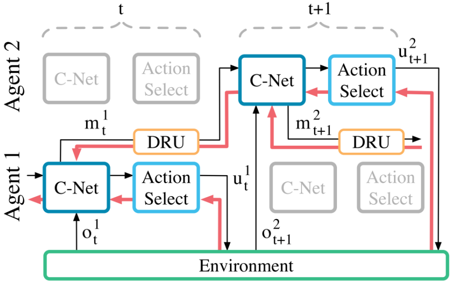
Multi-agent DDPG [7]

- Train in centralised manner, i.e. $Q_i^{\mu}(o_1, a_1, \ldots, o_N, a_N)$
- Execute in decentralised manner, i.e. $\mu_i(o_i)$
- If agents know the actions taken by other agents, the environment is stationary even when any $\mu_i \neq \mu'_i$
  $\mathbb{P}(o'_i | o_i, a_1, \ldots, a_N, \mu_1, \ldots, \mu_N) = \mathbb{P}(o'_i | o_i, a_1, \ldots, a_N, \mu'_1, \ldots, \mu'_N)$
- During training, agents learn coordinated behaviours
- In execution time, each agent acts according to its own learnt coordinated behaviour without any explicit communication



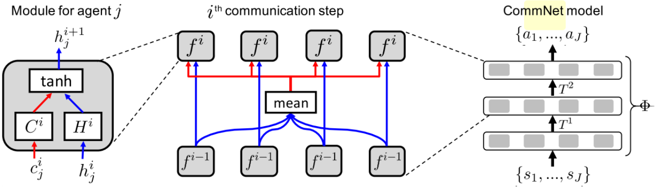
Communication-based Approaches

- Differentiable Inter-Agent Learning (DIAL) [8]
  - Sends gradients through the communication channel
  - 1-bit discrete messages
  - Weight sharing
  - $(N-1)N$ transmissions are required
  - Considers problems that can be solved with yes/no type of communication
  - Hard to scale to harder problems as gradients pass between the agents



Communication-based Approaches

- Communication Neural Net (CommNet) [9]
  - Sends gradients through the communication channel
  - Communication channel carries the average of the messages of all agents
  - Weight sharing
  - Uses a large single network for all the agents, which may not be easily scalable



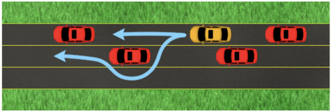
Outline

1. Reinforcement Learning [1]
   - RL in a Nutshell
   - RL Basics
   - Deep Reinforcement Learning

2. Multi-agent Reinforcement Learning
   - MARL Basics
   - Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

POMG with Extremely Noisy Observations

- In regular POMG, each agent receives a private partial observation o_i^t correlated with s^t
- What if some partial observations are extremely noisy, almost uncorrelated with s^t?
- Real-life example:
  Autonomous driving?



- Can existing solutions solve this problem?
  - DDPG: Over-over-optimistic
  - MADDPG: Is coordinated behaviour enough?
  - DIAL: Needs more than yes/no communication.
  - CommNet: If the majority is noisy, then how good could be the average of the observations?
  - Meta-agent: Can it learn to suppress the noisy info?

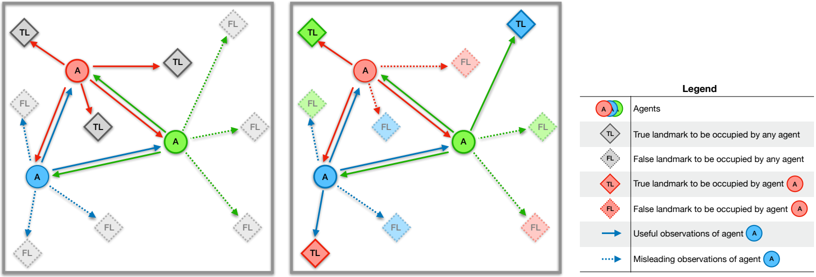
Environment: OpenAI Gym's Original Spread Scenario [7]

- N agents need to learn to reach N landmarks while avoiding collisions with each other
- Observations:
  - Their own positions and velocities
  - Relative positions of the other $N-1$ agents
  - Relative positions of the N landmarks
- Rewards:
  - Collective reward based on their distance to the landmarks
  - Additional negative reward if they collide with each other
- Actions:
  - Continuous N, W, S, E

Environment: Our Modified Spread Scenarios

- N agents need to learn to reach N landmarks while avoiding collisions with each other
- Observations:
  - Their own positions and velocities
  - Relative positions of the other $N-1$ agents
  - ~~Relative positions of the N landmarks~~
- Rewards:
  - Collective reward based on their distance to the landmarks
  - Additional negative reward if they collide with each other
- Actions:
  - Continuous N, W, S, E

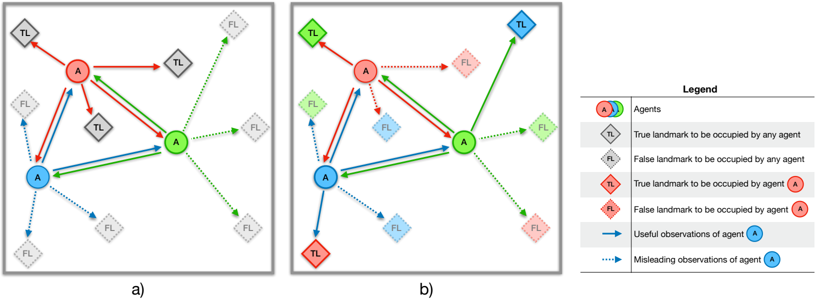
Environment: Our Modified Spread Scenarios



a) b)

- a) *Broadcasting* (one-to-all)
    - Any agent can occupy any landmark (as long as it is true)
    - The *gifted* agent is able to correctly observe all three landmarks
    - The other agents receive the wrong landmarks' locations
    - This special agent can either remain the same throughout the whole learning period (*fixed*) or vary across episodes (*alternating*), and even within an episode (*dynamic*).

Environment: Our Modified Spread Scenarios



- b) *Unicasting* (one-to-one)
  - Each landmark is designated to a particular agent, and the agents get rewarded only when reaching their allocated landmarks
  - Each agent can only correctly observe one of the landmarks (either its own or another agent's)
  - Otherwise receives the wrong whereabouts of the remaining ones
  - Same variations: *fixed*, *alternating*, *dynamic*

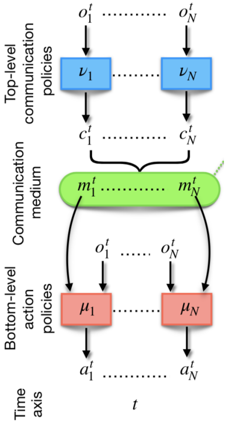
Proposed Approach

- Communication is a must
- Rather than sharing everything, filtering out the noisy observations may be advantageous
  - Resources, e.g. reduced communication cost, scalability
  - Performance?
- Agents with noisy information cannot discriminate between relevant and noisy information on their own
- Agents need to collectively decide which observations should be shared in the medium

Proposed Approach: Policies

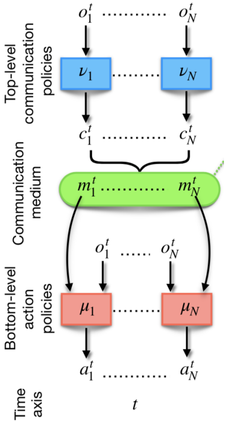
- Each agent has two hierarchically arranged policies $\boldsymbol{\nu}_i$ and $\boldsymbol{\mu}_i$
- $\{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_N\}$ and $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N\}$ are coupled through a communication medium $\{m_1, \ldots, m_N\}$

1. Each agent chooses a communication action c_i, i.e. $c_i = \boldsymbol{\nu}_i(o_i)$
2. $\{c_1, \ldots, c_N\}$ collectively determine the information shared in $\{m_1, \ldots, m_N\}$
3. Each agent determines its environmental action a_i, i.e. $a_i = \boldsymbol{\mu}_i(o_i, m_i)$



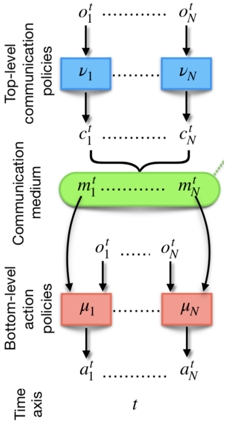
- The communication action of each agent is an N-dimensional vector, $c_{j,:} \in \mathbb{R}^{1\times N}$
- $c_{j,i}$ indicates the j^{th} agent's *willingness* to share its private observation o_j with the i^{th} agent
- The observation of the agent with the greatest *willingness* is shared with i^{th} agent, i.e. assigned to m_i

$$m_i = o_k \quad \text{where } k = \arg\max_j(c_{1,i}, \ldots, c_{j,i}, \ldots, c_{N,i})$$



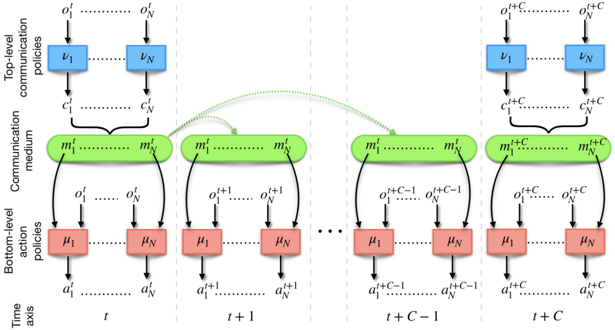
Proposed Approach: Training

- Problem: $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ are coupled and must be learned concurrently
- Use two different levels of temporal abstraction:
  - Run $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ at different frequencies
- Use different rewards to learn each policy:
  - Use cumulative of environment rewards to learn $\boldsymbol{\nu}$ and introduce some notion of auxiliary rewards to learn $\boldsymbol{\mu}$



Proposed Approach: Temporal Abstraction

- At time t, get c and determine m
- Keep c and m fixed for the next C steps, i.e. $m^t = \ldots = m^{t+C-1}$
- Obtain a for these C steps, $\{a^t, a^{t+1}, \ldots, a^{t+C-1}\}$, exploiting the information shared at time t



Proposed Approach: Rewards for Communication

- Recall that no explicit feedback for communication strategies
- Recall that environmental rewards indicates the distance to true landmarks
- To optimise communication policies $\boldsymbol{\nu}$
  - Recall that the c and m are fixed for C steps
  - Cumulative sum of the environmental rewards collected during these $C > 1$ steps may be a good feedback, i.e. $K_i = \sum_{t'=t}^{t+C} r_i^{t'}$
  - $C \to 1$: Is the received feedback due to communication actions or environmental actions?
  - $C \to T$: m would be too outdated

Proposed Approach: Rewards for Actions

- Recall that no explicit feedback for communication strategies
- Recall that environmental rewards indicates the distance to true landmarks
- To optimise action policies $\boldsymbol{\mu}$
  - Let's say we use environmental rewards to learn $\boldsymbol{\mu}$
  - When communication decisions are wrong, the observed rewards and the observations/actions will be uncorrelated
  - Instead, generate *medium-dependent rewards*, q, to motivate the agents to move towards the landmarks shared in the medium
  - Regardless of whether they are the noisy ones or the true ones

Proposed Approach: Learning

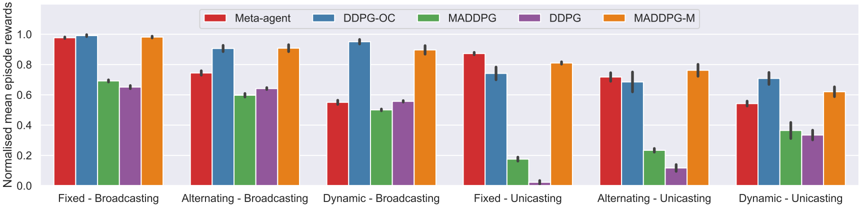
- Learning Q-values for communication policies $\boldsymbol{\nu}$, i.e. $Q^{\boldsymbol{\nu}}$, using K_i
  - $\mathcal{L}(\omega_{\nu,i}) = \mathbb{E}_{o,c,K,o''}\left[(Q_i^{\boldsymbol{\nu}}(o_1, c_1, \ldots, o_N, c_N) - y)^2\right]$
  - $y = K_i + \gamma Q_i^{\boldsymbol{\nu}'}(o_1'', c_1'', \ldots, o_N'', c_N'')\big|_{c_j''=\boldsymbol{\nu}_j'(o_j'')}$
- Learning $\boldsymbol{\nu}$ using $Q^{\boldsymbol{\nu}}$
  - $\nabla_{\theta_{\nu,i}} J(\boldsymbol{\nu}_i) = \mathbb{E}_{o,c\sim\mathcal{D}_\nu}\left[\nabla_{\theta_{\nu,i}}\boldsymbol{\nu}_i(o_i)\nabla_{c_i} Q_i^{\boldsymbol{\nu}}(o_1, c_1, \ldots, o_N, c_N)|_{c_i=\boldsymbol{\nu}_i(o_i)}\right]$
- Learning Q-values for action policies $\boldsymbol{\mu}$, i.e. $Q^{\boldsymbol{\mu}}$, using q_i
  - $\mathcal{L}(\omega_{\mu,i}) = \mathbb{E}_{o,m,a,q,o'}\left[(Q_i^{\boldsymbol{\mu}}(o_i, m_i, a_i) - y)^2\right]$
  - $y = q_i + \gamma Q_i^{\boldsymbol{\mu}'}(o_i', m_i, a_i')\big|_{a_i'=\boldsymbol{\mu}_i'(o_i', m_i)}$
- Learning $\boldsymbol{\mu}$ using $Q^{\boldsymbol{\mu}}$
  - $\nabla_{\theta\mu,i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{o,m,a\sim\mathcal{D}_\mu}\left[\nabla_{\theta_{\mu,i}}\boldsymbol{\mu}_i(o_i, m_i)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(o_i, m_i, a_i)|_{a_i=\boldsymbol{\mu}_i(o_i,m_i)}\right]$

Empirical Results: Comparison with Baselines

- DDPG: Ignore all MA problems and train agents independently
- MADDPG: Train in centralised manner, execute in decentralised manner. Can learn coordinated behaviour without any communication
- Meta-Agent: Use all available information, train and execute agents as a single agent with multiple control points. May be considered as a form of unlimited communication
- DDPG-OC: DDPG with Optimal Communication. Uses hard-coded optimal communication pattern
- MADDPG-M: Hierarchically learnt policies to filter out the noisy information

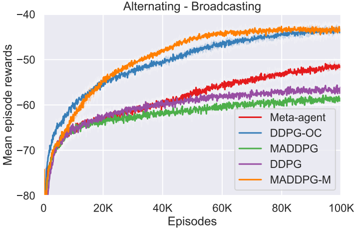
Empirical Results: Comparison with Baselines

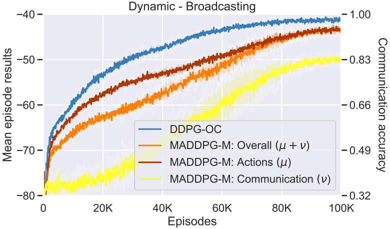
- DDPG and MADDPG: They both fail to learn the correct behaviour: Learning coordinated behaviour is not always helpful
- DDPG-OC: When m is optimally controlled, all the scenarios can be accomplished even by DDPG
- Meta-agent: Its performance decreases dramatically as the complexity of our environments increases: Using all available information is not always the best choice
- MADDPG-M: Performs quite similarly to DDPG-OC in all our environments: Underlying communication scheme as well as the optimal action policies can be learnt simultaneously by hierarchical training



Empirical Results: Simultaneous Learning of Policies

- In the initial phases of training, the MADDPG-M agents are able to begin learning the environment dynamics and the expected actions through the *medium-dependent rewards* rewards
- Improved environmental actions subsequently provide better feedback yielding improved communications actions, and so on
- Ultimately, MADDPG-M agents perform comparably to DDPG-OC





Conclusions

- A MARL problem characterised by partial and extremely noisy observations
- Two instances of this problem: *broadcasting* and *unicasting*
- The key technical contribution: hierarchical interpretation of the communication-action dependency
- Agents learn two policies that are connected through a communication medium
- Using different levels of temporal abstraction and intrinsically generated rewards
- We have considered scenarios where sharing a single observation at a time is sufficient to accomplish the task
- There might be more complex cases where an agent needs to reach the observations of multiple agents at the same time

Questions

Thank you for your attention!
Any questions?

References

D. Silver, "Reinforcement learning course notes,"

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction*.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pp. 157–163, 1994.

R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*.

J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*.

S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2244–2252, 2016.