

Multiagent Reinforcement Learning for Noised Communication in Fully Cooperative Environments

Jeffrey Li

*BSc Computer Science*

*Su*pervis*or:* Professor Mirco Musolesi

November 17, 2022

This report is submitted as part requirement for the BSc Degree in Computer Science at UCL. It is substantially the result of my work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

About MARL in general and tackling cooperation problem in noised environments

**Contents**

[1. Project Background 0](#_Toc130994727)

[2. Background Knowledge 0](#_Toc130994728)

[2.1 Markov Decision Processes 0](#_Toc130994729)

[2.2 Reinforcement Learning 1](#_Toc130994730)

[3. Multiagent Reinforcement Learning (MARL) 5](#_Toc130994731)

[3.1 An Introduction to MARL 5](#_Toc130994732)

[3.2 Challenges in MARL 6](#_Toc130994733)

[3.3 Modern MARL approaches 7](#_Toc130994734)

[4. Communication 8](#_Toc130994735)

[5. Related Work 10](#_Toc130994736)

[6. Problem Formulation 14](#_Toc130994737)

[7. Design and Implementation 14](#_Toc130994738)

[8. Numerical Results 14](#_Toc130994739)

[9. Conclusion 14](#_Toc130994740)

[10. Bibliography 14](#_Toc130994741)

# Project Background

Chapter Summary

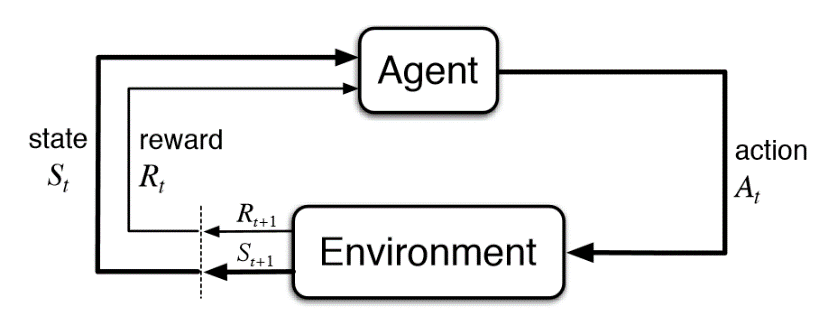
* TBD
* Centralised Training, decentralized execution MARL algorithm for agents to learn communication protocols to effectively communicate over noisy channels, such that agents can achieve coordination tasks in selected Multi-Particle Environments.
* Assumptions and constraints …
  + Partial observability
  + …

# Background Knowledge

This chapter aim to gently introduces the foundational underpinnings of all Reinforcement Learning problems, namely MDPs and the abstract Policy Iteration.

## 2.1 Markov Decision Processes

Markov Decision Processes (MDPs) (Howard, 1960) are classical formulations of sequential decision-making, which seek to abstract the problem of goal-directed learning from interactions. This set of frameworks consists of two components: the agent being the decision maker and the environment in which the agent interacts with.



*Figure 1. MDP Flow diagram (Sutton & Barto, 2018)*

At each timestep as shown in Figure 1, the agent would receive a representation of the environment's current state and a reward signal . Based on the current state, the agent selects an action where the set of available actions to choose from depends on the current state . In response to the action, at the next time step , the agent would receive a real number reward and would be in a new state . There is a probability of those values occurring at time , given values of current state and action:

This probability characterizes the dynamics of the environment and also satisfies the law of total probability for each choice of and :

From this, we could also observe that the probability of each value of and depend entirely on the state and the action of the agent one timestep before. In other words, Markov Property states that the future depends completely on the present and not on the past, and any scenario which follows this property is defined as Markovian.

## 2.2 About Reinforcement Learning

### 2.2.1 Problem Setting

Reinforcement Learning (RL) solves problems that has an underlying MDP framework, and the goal is the maximize the expected value of the return (cumulative sum of rewards). More precisely, it is maximizing the expected value of the discounted return with discount factor to consider continuous tasks which may have infinite .

(1)

This expected return is represented as state-value functions, which define the expected cumulative reward that an agent could receive at a particular state while following certain agent behaviours governed by what is known as policies. Policies are functions that map from state to the probability of selecting each action , denoted as . Thus the state-value function at state while following policy has the following definition:

(2)

The value of a state can also be considered in conjunction with a specific action instead of an expectation over all possible actions, called action-value functions, which are more commonly used in many RL algorithms and can also be used to express state-value functions:

These value functions are unknown to the agent, and since the goal is to maximize the expected return, an RL problem can be formulated as estimating value functions given a policy and using that estimated value function as a basis for comparison to searching for the optimal policy that is better than any other policies.

Whether one policy is better than the other is defined as follows, which states that a policy is better than another policy if and only if the state-value function estimated given yields a larger or equal value to the state-value function estimated given at all states:

There may be more than one optimal policy, all of which are denoted as and share the same optimal state-value function :

And they also share the same optimal action-value function :

can also be used to express :

(3)

Any policy greedy to these optimal value functions is the optimal policy. This means, given the current state , for , a policy that always selects the action which leads to the highest , and for a policy selecting an action that leads to the highest q value.

### 2.2.2 Bellman’s Equations

Bellman’s Equation is used to express mathematically:

*From the recursive definition of (1)*

*From definition of (2)*

being a state-value function of optimal policies must satisfy this condition, but also, it can be expressed without reference to any policy, and this equation is known as Bellman’s optimality equation:

*By recursive definition of (1)*

*Optimal expected return (2)*

And solving this equation would provide one way to find the optimal policy, but doing so is rarely useful in practice as solving this optimality equation is equivalent to an exhaustive search over all possibilities and requires a large amount of computational power. Also as will be explained in the following sections, the dynamics of the environment are rarely known by the agent to begin with.

### 2.2.3 Tabular RL

Model-based and model-free

Model-Based algorithms as suggested by their name requires knowledge of the model of the environment in order to find the optimal policy. In RL's case, the model refers to knowing the dynamics of the environment. Recall that the dynamics of an environment are categorized by the following probability:

Some well-known Model-Based algorithms are the Value iteration and Policy iteration

In practice, the dynamics of the environment is rarely known and a model would not be readily available for the agent at the beginning, hence these algorithms are rarely useful.

For Model-Free algorithms, the dynamics of the environment is unknown to the agent, and hence … This is

* Monte Carlo
* Temporal Difference Learning
* Q-learning(Watkins & Dayan, 1992)
  + Off Policy

### 2.3.3 Deep Reinforcement Learning

As shown in the Q-Learning method, the

To account for larger state spaces, RL algorithms using function estimators became increasingly popular, one such method involves using neural networks.

* Successes in DRL
  + Mastering the game of Go with deep neural networks and tree search.
* Deep Q Networks (DQL) (Mnih et al., 2015)
  + Use of neural network
  + Experience Replay

Policy gradient

* General Policy Gradient
  + <https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html>
  + Known to exhibit high variance gradient estimations
    - More so in a multiagent context as the agent’s reward depends on the action of many agents, therefore reward conditioned only on the agent’s own actions exhibits higher variance
* Actor-Critic
  + Rather similar to GAN
  + Temporal Difference version of policy gradient
  + Contains an actor and a critic
    - The actor decides which action should be taken
    - The critic informs the actor how good was the action and how it should adjust
  + The actor learns through the policy gradient approach; critics evaluate the action produced by the actor by computing the value function

# Multiagent Reinforcement Learning (MARL)

Chapter Summary

## 3.1 An Introduction to MARL

From the surface, it seems that Multiagent Reinforcement Learning is solving similar MDPs as RL but in a multi-agent setting known as Markov games (Littman, 1994).

More on explaining the multi-agent MDP settings…

Unlike single-agent reinforcement learning, which can only interact with its environment, in the multi-agent setting, agents would in addition be able to interact with other agents. This leads to a broader type of interaction, namely competing where all agents seek to accomplish a goal over the other agents, cooperating where agents would work together to achieve a common goal or some mixture of the both. Therefore, this setting inherently can be applied to more interesting applications such as …

Many interesting applications:

* such as the AI Economist(Zheng et al., 2020) which aims to improve equality and productivity with AI-Driven Tax policies,
* the study of sequential social dilemmas which require multiple agents to learn policies that implement their strategic intentions (Leibo et al., 2017)
* Multiagent RL applied in playing video games(Peng et al., 2017)

Cooperative AI(Dafoe et al., 2021) especially has gained increasing attention over the years and radiates through a wide range of impactful fields, and the study of reputation in cooperative systems(Anastassacos et al., 2021) to name a few.

* OpenAI agents playing hide and seek(Baker et al., 2020)

Robotic applications

* This is a very interesting problem to tackle that can be useful in practical applications such as setting up protocols for multiple robots to do security patrolling

## 3.2 Challenges in MARL

A widely used algorithm to solve such cooperative problems in the multiagent context is known as the Independent Q-Learning (IQL) algorithm. This algorithm trains all the agents with Q-Learning individually and each agent would learn their own Q function that only conditions on its own states and actions, this makes it a decentralised learning and execution algorithm. However, this method of learning has been shown to perform poorly in multi-agent settings (Matignon et al., 2012). The is because, in the multi-agent context, the policy of the individually trained agents changes at training, which causes non-stationarity of the environment in other agents’ perspectives since the agents themselves are also part of the environment.

This is one of the major challenges not only for IQL but also for MARL in general as the non-stationarity in agent’s observations breaks the Markov assumption that governs convergence of Q-Learning, of which agents would end up in an infinite loop of adapting to other agent’s policies. This inherent property of MARL problems also prevents naïve approaches of experience replay, since non-stationarity of stored states defeats the purpose of having a replay memory in the first place that is meant to stabilize training of deep networks, and more sophisticated approaches are required.

Some attempts in amending the Independent Q Learning involved tackling the non-stationarity issue such as inputting other agents’ policies as a parameter to the Q function of the agent being trained to overcome the non-stationarity (Tesauro, 2003); hysteretic Q Learning(Matignon et al., 2007); conditioning the agent’s value function on a fingerprint that disambiguates the age of the data sampled from the replay memory, essentially indexing the experiences to allow experience replay to work (J. Foerster, Nardelli, et al., 2018), but there exist more sophisticated and robust approaches.

## 3.3 Modern MARL approaches

Approaches to tackle non-stationarity(Papoudakis et al., 2019):

* Centralised critic approaches the most popular approach, a lot of results
  + COMA(J. Foerster, Farquhar, et al., 2018)
    - Counterfactual multi-agent policy gradients, uses centralised critic to estimate Q function and decentralised actors to optimise agents’ policies
    - Address the challenges of multi-agent credit assignment, uses a counterfactual baseline that marginalised out a single agent’s action while keeping the other agents’ actions fixed
    - Learns a single centralized critic for all agents
  + MADDPG(Lowe et al., 2020)
    - Actor critic policy gradient where the critic is augmented with extra information about the policies of other agents, while the actor only has access to local information. After training is completed, only the local actors are used at execution phase
    - Since the centralized critic function explicitly uses the decision-making policies of other agents, they showed that agents can learn approximate models of other agents online and effectively use them in their own policy learning procedure.
    - acting in a decentralized manner and equally applicable in cooperative and competitive settings.
    - learn a centralized critic for each agent, allowing for agents with differing reward functions including competitive scenarios
    - consider environments with explicit communication between agents
    - learns continuous policies
  + Extend policy gradient framework to Deterministic Policy Gradient (DPG) algorithms
    - Deep DPG is a variant of DPG where the policy and critic are approximated with deep neural networks
      * <https://arxiv.org/pdf/1509.02971.pdf?source=post_page--------------------------->
      * Also make use of experience replay as in DQN to stabilize the neural network
* Decentralised Learning techniques
* Opponent modelling
* Meta-Learning
* Communication

Some other algorithms

* VDN
* QMIX
* MAPPO(Yu et al., 2022)

# Communication

Chapter Summary

Communication is very useful for MARL problems as it allows agents to exchange information about their states and actions which helps to stabilize training.

* K.-C. Jim and L. Giles, “How communication can improve the performance of multi-agent systems,” in Fifth Int’l Conf.  
  on Autonomous Agents, AGENTS ’01, (New York, NY), p. 584–591, 2001.
* T. Balch and R. Arkin, “Communication in reactive multiagent robotic systems,” Autonomous Robots, pp. 27–52, 1994.

Three levels to communication(Shannon & Weaver, 1964)

* They described level A as the technical problem,  
  which tries to answer the question “How accurately can the symbols of communication be  
  transmitted?”. Level B is referred to as the semantic problem, and asks the question “How  
  precisely do the transmitted symbols convey the desired meaning?”. Finally, Level C, called  
  the effectiveness problem, strives to answer the question “How effectively does the received  
  meaning affect conduct in the desired way?”.

This problem involves many different dimensions(Zhu et al., 2022):

* Communicatee type (Do agents communicate directly with each other or through a proxy)
  + Proxy (an agent that only act as communication medium, with no interaction with the environment)
  + Agents in the system
* Communication policy
  + Full communication
  + Partial structure
  + Individual control
  + Global control
* Communicated Messages
  + Existing Knowledge
  + Imagined future knowledge
* Message combination
  + Concatenation
  + Equally valued
  + Unequally valued
* Inner integration
  + Policy level
  + Value level
  + Policy level and value level
* Constraints
  + Limited bandwidth
  + Noise
  + Shared communication medium (not really related in coop case)
* Communication Learning
  + Reinforcement Learning
  + Differentiable
* Training scheme
  + Decentralised training
  + CTDE
* Goal
  + Coop
  + Competitive
  + Mixed

With these dimensions, I would classify my work as …

# Related Work

Chapter Summary

This project focuses on communication in MARL context to establish efficient cooperation.

Deciding when to communicate

* Individualized Controlled Continuous Communication Model (IC3Net)(Singh et al., 2018)
  + Extended from CommNet
  + Controls communication with gating mechanism to decide when to communicate and uses individualized rewards for each agent to gain better performance and scalability
  + Applied to cooperative, semi-cooperative and competitive settings
* <https://arxiv.org/pdf/1902.01554.pdf>
  + SchedNet
  + Limited bandwidth
  + the agents share the communication medium so that only a restricted number of agents are able to simultaneously use the medium
    - to simulate state-of-the-art wireless network architectures
  + Learns to decide which agents should be entitled to  
    broadcasting their (encoded) messages, by learning the importance of each agent’s  
    partially observed information.

BiCNet

My project aims to tackle the communication protocol learning dimension within the bigger picture of cooperative MARL communication under the constraint of having a noised communication channel.

Some earliest attempts involved the study of synthetic ethology(MacLennan & Burghardt, 1993) that investigated the mechanisms and evolution of communication in finite state machines using genetic algorithms to learn to cooperate in a simplified environment. Such approach is also adopted in allowing predator agents to learn to communicate in the grid-world predator-prey problem(Giles & Jim, 2003), hence allowing these antagonistic agents to coordinate with each other for more efficient capture of the prey. However, these genetic algorithm approaches would not be scalable for larger problems.

As identified in the very recent survey(Zhu et al., 2022) , it is commonly accepted by the modern research community in this field that there are two paradigms to learning a communication protocol, one that assumes continuous property of the communication channel and applies backpropagation through gradient descent for optimizing, and the other treating the channel as an extension to action space, and the communication protocol is learned through standard RL algorithms (Reinforced Communication Learning).

For the backpropagation approach, it has been discovered that it tends to converge quicker to better policies compared to the other framework, due to not needing any actual interactions with the message channel space. The state-of-the-art algorithm DIAL(J. N. Foerster et al., 2016) is a centralised training and decentralised execution algorithm, which allows combination of centralised learning and deep Q networks achieved by pushing gradients through the communication channel from one agent to another, hence taking full advantage of centralised learning.

* Share parameters
* Because DIAL passes gradients from agent to agent, it is an inherently deep learning approach.

Another well-known state-of-the-art algorithm CommNet(Sukhbaatar et al., 2016) …

* Learns a shared Deep Neural Net that is shared across agents
* Shared reward

On the other hand, the other approach is more commonly applied to problems with the assumption that communication channels are discrete, meaning it would not be differentiable for backpropagation techniques to be applied for optimization. Thus, optimization of communication protocol is achieved via applying RL algorithms. The assumption is closer to real life …

Within this paradigm, some early algorithms were developed using the tabular Q Learning method to learn set of communication codes/protocols to solve the predator-prey problem(Kasai et al., 2008). Unfortunately, as explained in previous sections, these approaches are not scalable due to the nature of tabular storage. A better approach is the RIAL(J. N. Foerster et al., 2016) algorithm which is introduced in the same paper as DIAL that make use of Deep Q Learning to overcome this issue. RIAL combines DRQN with independent Q-Learning …

* RIAL method makes use of deep Q Learning for better scalability to learn content of the message
* Share parameters

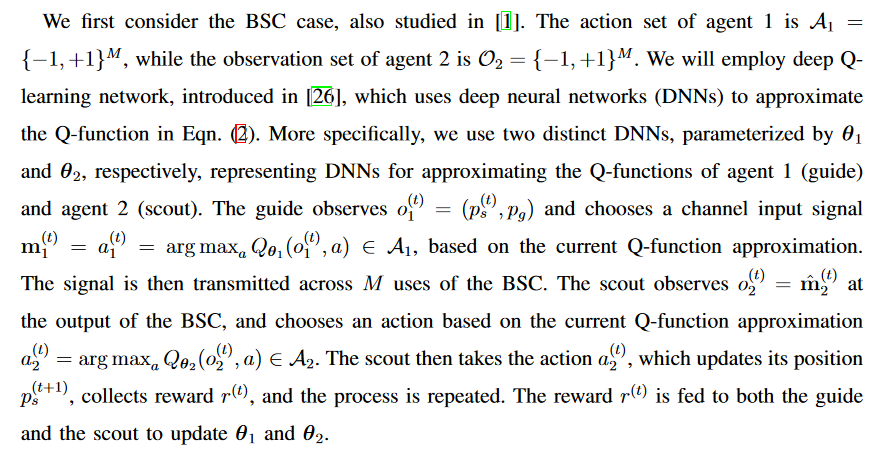
Another algorithm that is worth mentioning is the DDRQN algorithm(Kasai et al., 2008)…

* Discover communication protocols to solve multiagent learning problems based on well-known riddles (partially observable tasks)
* Not really solving coordination problems

However, most algorithms do not consider the settings where the communication channel is noised. The algorithm that is most similar to my approach is the DiffDiscrete algorithm(Freed et al., 2020). This algorithm seeks to learn a communication protocol on a discrete communication channel with additive noise that is unknown to the agents. But instead of using the Reinforced Communication Learning approach mentioned before, the algorithm uses a stochastic message encoding/decoding procedure that makes a discrete communication channel mathematically equivalent to an analogue channel with additive noise, then which gradients can be backpropagated for optimization. The algorithm uses … to tackle the noise issue

What distinct my approach from DiffDiscrete is that …

Another very recent work that is worth mentioning is (Tung et al., 2021)

* Might want to swap around the order
* Joint channel coding and modulation
* As
* Different problem
  + Cooperation between the single guide and single scout
  + 
    - Both Guide and scout are DQN agents
    - Whereas for my work, I focus on cooperation between the scouts, guide’s only purpose is to distribute its observations to the scouts
* Interestingly proposed the guide-scout scenario, which I would adopt my algorithm on
* Limitations
  + One guide one scout scenario
  + Not robust, agent need to be re-trained for each noised scenario

# Problem Formulation

Chapter Summary

# Design and Implementation

Chapter Summary

# Numerical Results

Chapter Summary

# Conclusion

# Bibliography

Automatic citation updates are disabled. To see the bibliography, click Refresh in the Zotero tab.