Critique of Learning to Communicate with Deep Multi-Agent Reinforcement Learning

Ashutosh Kumar Jha, ME14B148
Mechanical Engineering, IIT Madras
Topics in RL Course

***Abstract*—In this technical report, we summarize the contents and ideas presented in the technical paper titled** *Learning to Communicate with Deep Multi-Agent Reinforcement Learning*[1]**. We also present the possible shortcomings of the work.**

Keywords*—*Deep Reinforcement Learning, Multi-Agent Reinforcement Learning

I. Critique

The paper starts by considering an Artificial General Intelligence question of how language and communication truly emerge between intelligent agents. The authors claim that they try to answer these questions through this work by looking at the Multi-Agent RL setting where the agents are composed of Deep Neural Networks.

The critique is structured as follows: We first present the problem setting. Then, we present the background necessary(assuming the person reading this critique understands DQNs) for understanding the experiments. Once the background is provided, we present the methods presented by the paper. After that, the experiments and results of the paper are presented. In the next section, we present the shortcomings of the work.

A. Problem Setting:

The paper considers the Reinforcement Learning setup to start with. The following additional constructs are added to the basic RL setup:

1) **Multi-Agent RL:** This refers to the fact that instead of a single agent, there are multiple agents acting in the same environment.
2) **Partial Observability:** No agent can observe the underlying Markov state s_t. Instead each agent(denoted by a) receives a observation o_t^a correlated to s_t.
3) All agent act during a given time-step. In each time-step, each of the agents selects an environment action $u \in U$ that affects the environment, and a communication action $m \in M$ that is observed by other agents but has no direct impact on the environment or reward.
4) **No communication protocol is provided** to the agents a priori. As a result, the agents must develop and agree upon such a protocol.

The authors justify the problem setting with the original goal(of learning about language) they set up for the paper: it is only when multiple agents and partial observability coexist that agents have the incentive to communicate.

The paper focuses on problem settings with centralised learning but decentralised execution. That is, communication is not restricted between agents during learning however during execution of the learned policies, the agents can communicate only via limited-bandwidth channels.

B. Background

1) Independent DQNs: DQNs have been extended to cooperative multi-agent settings by [2]. In independent Q-learning, each agent independently learns its own Q-function $Q^a(s, u^a; \theta_i^a)$. All the agents act simultaneously in the environment and receive a team reward. A schematic of the Independent DQN is shown in Figure 1

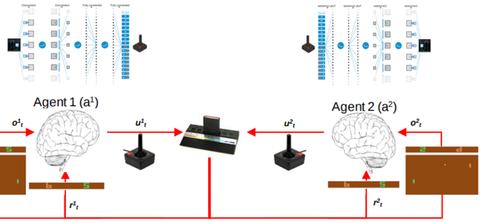


Fig. 1. Schematic of Independent DQN

2) Deep Recurrent Q-Networks: DQNs were originally designed to solve a fully observable MDP. While solving POMDPs however, a history of observations is known to help in solving the POMDP. To add this ”chronological memory” to the DQN, Hausknecht and Stone [3] proposed Deep Recurrent Q-Networks. The idea is to use a recurrent neural network instead of a feed forward neural network as the function approximator for the action value function. Thus, the action value function is now also a function of the previous state of the RNN in addition to the observation and action. A schematic of the DRQN is provided in Figure 2.

C. Methods Proposed

The paper presents two methods for learning the protocols of communication between the agents. They first propose a naive method called *Reinforced Inter-Agent Learning(RIAL)* and then propose their method *Differentiable Inter-Agent Learning(DIAL)* as an extension to RIAL.

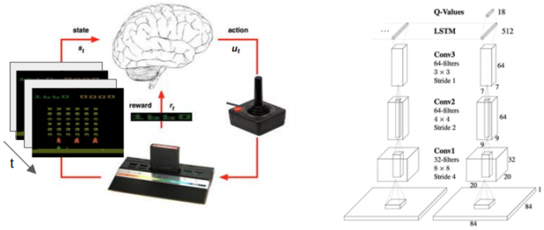


Fig. 2. Schematic of Deep Recurrent Q-Network

1) Reinforced Inter-Agent Learning: The idea behind RIAL is simple: combine the ideas of Independent Q-Learning and DRQNs for action **and** communication selection. Each agent thus now outputs $Q^a(o_t^a, u_t^a)$ and $Q^a(o_t^a, m_t^a)$.

In the RIAL agent the network is split to output the Q values for the action and the communication selection instead of outputing a single Q-value to reduce the number of output heads.

Two modifications were made to the DQN framework. First, the experience replay was disabled in order to stabilize training in a multi-agent setting. Second, to account for partial observability, actions u and m taken by each agent are fed as inputs on the next time-step.

So far, the RIAL setup doesn't really make use of the centralised training phase as the gradients don't flow across the agents. To take advantage of the centralised training phase, the agents are made to **share their parameters**, thus training a single network. The agents can still behave differently because they receive different observations and thus evolve different hidden states. In addition, each agent receives its own index a as input, allowing them to specialise.
The RIAL Setup along with the backpropogating flow of gradients is shown in Figure 3

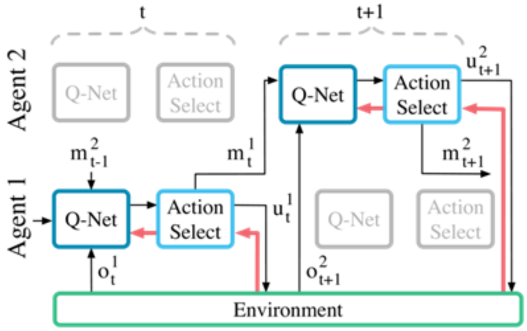


Fig. 3. Schematic of RIAL

2) Differentiable Inter-Agent Learning: As mentioned earlier, RIAL is really a naive framework in the sense that there is no gradient flow **across** agents. In other words, the agents do not give *each other* feedback about their communication actions.

The main insight behind DIAL is that the combination of centralised learning and Q-networks makes it possible, not only to share parameters but to push gradients from one agent to another through the communication channel. DIAL is thus end-to-end trainable across the agents.

In the DIAL centralised learning phase, the agents send across real-valued messages to other agents so that gradients can be taken over messages. During execution phase however, since only discrete valued messages are allowed, a simple discretization is performed on the message. The trick is simple: The messages in a channel is represented by using a sigmoid over a gaussian. This tends to push the values close to the tails of the sigmoid which approximates discretization during training as well. The paper call this as the **Discretizing/Regularizing Unit(DRU)**.
The schematic of the DIAL framework is shown in Figure 4.

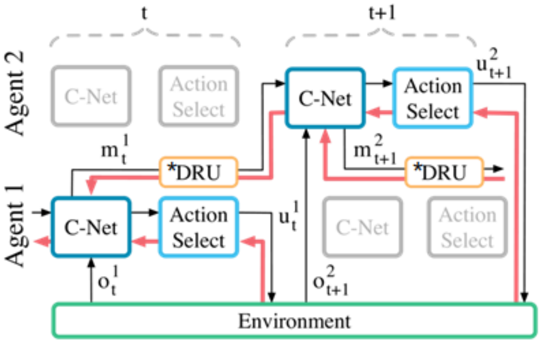


Fig. 4. Schematic of DIAL Framework

Model Architecture:: RIAL and DIAL both share the same individual model architecture. each agent consists of a RNN, unrolled for T time-steps, that maintains an internal state h, an input network for producing a task embedding z, and an output network for the Q-values and the messages m. The input for agent a is defined as a tuple of $(o_t^a, m_{t-1}^a, u_{t-1}^a, a)$.

The task embedding is defined as:

$$z_t^a = (\text{TaskMLP}(o_t^a) + \text{MLP}[|M|, 128](m_{t-1}) \\ + \text{Lookup}(u_{t-1}^a) + \text{Lookup}(a))$$

Where TaskMLP, MLP and Lookup can be replaced by other function approximators. A batch normalization layer was used to preprocess m_{t-1}. The full architecture is shown in Figure 5.

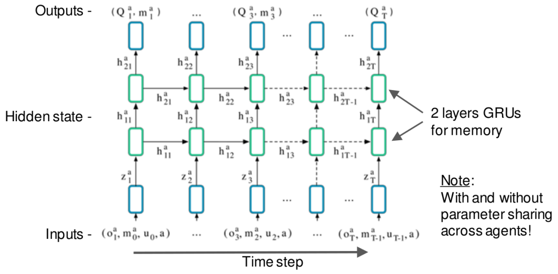


Fig. 5. Model Architecture for DIAL and RIAL

D. Experiments and Results

The paper shows results on three different tasks. The first is a switch riddle from psychology and the other two tasks are based on MNIST. In the first tasks, N different prisoners such that every day one prisoner gets sent to the interrogation room where he sees the switch and chooses from On, Off, Tell and None. The game is won when the ”Tell” action is taken when all prisoners have visited the room.

In the MNIST games, there are two agents and each is shown a number. The goal is to be able to predict the other agent's number completely. The first MNIST game is a Color Digit task. Here, a number with a color is shown to both the agents. Only one bit can be communicated. The reward is designed so that communicating parity would give maximum reward (the agents are supposed to learn this). In the second task Multi-Step MNIST, a gray scale digit is supposed to be predicted when 4 bits are communicated, thus allowing agents to learn a code for each of the 10 digits.

The following agents are compared: The full RIAL agent, The full DIAL agent, The non-parameter sharing versions of the DIAL and RIAL agents and an agent where no communication is allowed.

In both the experiments, the full DIAL agent outperforms the other agents by a margin with the exception of the Switch riddle with 4 prisoners where RIAL performs slightly better. Once the agents are trained, their protocols and policies can be visualized in the execution phase. The policy for the switch riddle for 3 prisoners is shown in Figure 6. The protocol for the Multi-Step MNIST is shown in Figure 7. As expected, the agents learn a code for each of the digits.

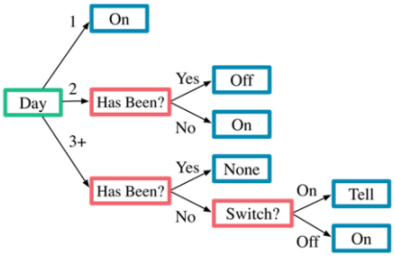
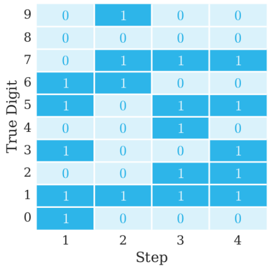


Fig. 6. The Strategy for 3 prisoners case



True Digit	1	2	3	4
9	0	1	0	0
8	0	0	0	0
7	0	1	1	1
6	1	1	0	0
5	1	0	1	1
4	0	0	1	0
3	1	0	0	1
2	0	0	1	1
1	1	1	1	1
0	1	0	0	0

Fig. 7. Protocol learned for Multi-Step MNIST

II. Shortcomings

In this section we make a note of possible short comings of this paper:

- The results are on really low dimensional data. I have my doubts about the scalability of the methods proposed.
- The paper only explores cooperative settings in Multi-Agent RL. Will be interesting to see applications in Non-Cooperative settings.
- Not exactly a shortcoming, but a possible direction of future work is to look at how communication can be improved in the presence of a *novice* agent in the environment.

III. Conclusion

The paper is the first paper to look at Multi-Agent RL in a setting where the bandwidth of communication is limited. Personally, I liked the analysis of the results however, I would love to see the scaling of the paper to higher dimensional games (something like Soccer or Counter-Strike). All in all, a very interesting read in terms of the implications it has on General Artificial Intelligence in terms of emergence of language.

References

[1] Learning to Communicate with Deep Multi-Agent Reinforcement Learning, Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, Shimon Whiteson Advances in Neural Information Processing Systems(NIPS), 2016.

[2] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. arXiv preprint arXiv:1511.08779, 2015.

[3] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. arXiv preprint arXiv:1507.06527, 2015.