Learning to Communicate in Multi-Agent Settings

Joel Oskarsson
Linköping University
joeos014@student.liu.se

Abstract

Communication is neccesary for groups of agents to properly coordinate actions. When agents learn coordinated strategies it is often desirable that a suitable communication protocol is learned simultaneously. This report gives an overview of different methods proposed for multi-agent communication learning. A number of different models and learning algorithms are presented, followed by a discussion about applications and future work.

1 Introduction

Communication and language are central parts of what we consider human intelligence. The ability to use communication to coordinate efforts is integral to much of modern society. Communication, although between computers, has also been at the centre of the digital transformation of society.

Fueled by developments in the general machine learning area, and deep learning in particular, new approaches have been proposed for multi-agent learning. These approaches balance centralized learning with learning distributed over multiple agents [Foerster *et al.*, 2018; Lowe *et al.*, 2017]. Both cooperative and competitive settings have been considered.

A natural continuation on the work in multi-agent learning is to introduce the ability for agents to communicate. This should improve coordination and lead to better behaviors. Research in this area might also lead to further understanding of how human and animal communication emerges [Foerster *et al.*, 2016]. A number of approaches for learning communication protocols between multiple agents have been proposed in the literature. This report aims to give a short survey over the area of communication learning and discuss some applications and future work.

2 Theory

The following sections give an introduction to the setting and learning algorithms used in communication learning.

2.1 Reinforcement Learning

Most of the proposed approaches to communication learning build on the framework of Reinforcement Learning (RL). In the single agent case, RL models an agent choosing actions to carry out in some environment [Sutton and Barto, 1998]. This can be formally modeled by a Markov Decision Process (MDP), defined as a 4–tuple $(\mathcal{S}, \mathcal{A}, p, R)$. $\mathcal{S}$ is the set of possible states the environment can be in. $\mathcal{A}$ is the set of actions available to the agent. p is a function determining the dynamics of the environment, that is which state transitions take place as consequences of agent actions. Only model-free settings, where p is initially unknown to the agent, will be considered. R is a reward function, determining the reward signal sent to the agent based on states and actions. In the most general formulations both state transitions and rewards can be stochastic, described by probability distributions.

This framework can easily be extended to a setting with N agents [Lowe *et al.*, 2017]. In this case there is one action space per agent $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N$ and p takes the actions of all agents into account. The agents also receive individual rewards, extending R to N reward functions $R_1, R_2, \ldots, R_N$.

In communication learning the state is often not fully observable by the agents, extending the setting to a Partially Observable MDP (POMDP) [Das *et al.*, 2019]. In this case the agents do not observe the environment state $S \in \mathcal{S}$ directly. Instead each agent receives an observation ω_i from its observation space $\mathcal{O}_i$. ω_i is correlated with the full state S. The partially observable setting is of particular interest when considering communicating agents. By communication the agents can combine information from all observations $\omega_1, \omega_2, \ldots, \omega_N$ to form a more complete belief about S [Das *et al.*, 2019].

2.2 Algorithms

The goal of RL in POMDPs is for each agent to learn a policy $\pi_i(a_i|\omega_i)$, describing how it should act based on its given observation [Sutton and Barto, 1998]. The best policy maximizes the expected reward the agent receives. There are many different ways to learn such a policy.

Policy Gradient Methods
Many of the communication models proposed make use of policy gradient methods [Sutton and Barto, 1998] with neural networks as function approximators. These methods directly parametrize the policy π_i as a neural network, with the output of the network describing a probability distribution over $\mathcal{A}_i$. The problem of searching for a good policy then reduces to finding a set of network parameters θ_i that makes the policy $\pi_{\theta_i}(a_i|\omega_i)$ achieve high expected reward.

Q-Learning
Another method that has found use in multi-agent settings is Q-learning [Kasai *et al.*, 2008; Foerster *et al.*, 2016]. The Q-function is defined as the expected future reward when the agent follows policy π:

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^k R(S_{t+k+1}) \middle| S_t = s, A_t = a \right] \quad (1)$$

where γ is a discount factor, determining the value of future rewards [Sutton and Barto, 1998]. In traditional Q-learning a technique called temporal-difference learning [Sutton and Barto, 1998] is used to estimate $Q_\pi(s,a)$. In Deep Q-learning $Q_\pi(s,a)$ is estimated using a neural network, referred to as the Deep Q-Network (DQN). The policy $\pi(a|s)$ for a state s can then be retrieved by choosing the action a that maximizes $Q_\pi(s,a)$. Q-learning can readily be extended to the POMDP setting [Egorov, 2015].

3 Learning to Communicate

The problem of learning to communicate extends the POMDP setting by providing a communication interface between the agents. This complicates the learning task by incentivizing the agents to:

1. Condition their actions on messages received
2. Learn what messages to send

A useful distinction can be made between methods using discrete or continuous messages.

The following sections consider settings where agents are trained to collaborate. This is generally achieved by letting all agents have the same reward function R and thus receiving the same reward.

3.1 Discrete Messages

When learning communication with discrete messages the set of possible messages $\mathcal{M}$ is restricted to some finite set. A typical choice is $\mathcal{M} = \{0,1\}^L$, the set of binary vectors of some fixed length L [Kasai *et al.*, 2008; Foerster *et al.*, 2016].

Q-learning
Kasai *et al.* [2008] consider two agents communicating with binary messages of length $L \leq 4$. Their experiments make use of traditional Q-learning, estimating separate Q-values for state-action pairs and state-message pairs. An action policy is then derived from $Q_a(s_i, a_i)$ and a messaging policy is derived from $Q_m(s_i, m_i)$. The messaging policy defines which message to send in each state. Traditional Q-learning does not scale well to large environments [Lin, 1993], but is possible here due to the small size of the state, action and message spaces.

RIAL
Foerster *et al.* [2016] extend the work on discrete message communication with the use of deep Q-learning. They introduce a communication learning method called Reinforced Inter-Agent Learning (RIAL). Similarly to the work by Kasai *et al.* [2008] RIAL uses separate Q-values for actions and messages. In RIAL these Q-values are however estimated using DQN. Foerster *et al.* [2016] also extend the DQN method by sharing parameters between all agents' networks. To give the agents some notion of memory throughout each episode Recurrent Neural Networks (RNNs) are used for the Q-networks.

In their experiments Foerster *et al.* [2016] try evaluating RIAL on a digit communication task. Two agents are shown individual images of digits from the MNIST dataset[1] and are supposed to communicate their digit to the other agent. In this test the RIAL method fails to learn useful communication. The authors hypothesize that this is because the agents fail to discover that the communication can be useful. In light of this result Eccles *et al.* [2019] in a later paper propose a modified learning method for RIAL. By changing the minimized loss a bias is introduced for both sending useful messages and for listening to other agents. It is shown that this modification allows RIAL-agents to successfully communicate digits.

DIAL
One of the shortcomings of the RIAL method is the weak connection between actions and messages. Foerster *et al.* [2016] propose a way to get around this through an extension called Differentiable Inter-Agent Learning (DIAL). DIAL only uses DQN for selecting actions and lets the messaging parts of the model be trained implicitly. This is possible since the actions selected depend on any messages received.

To understand further why this approach works and why it is not applicable to RIAL, recall how neural networks are trained. The Q-network needs to be trained using gradient descent [Mnih *et al.*, 2015]. A consequence of this is that gradients are calculated with respect to all trainable parameters. Discretization, such as for the messages in RIAL, results in gradients being either undefined or equal to 0. DIAL gets around this by introducing a Discretize/Regularize Unit (DRU).

Let $\hat{m}_i^t \in \mathbb{R}^L$ be a specific message output from agent i's network at time t. The DRU is applied to this real-valued vector to get the final message $m_i^t = \text{DRU}(\hat{m}_i^t)$ that is sent to other agents. The DRU has different behaviours during training and testing. Its effects are ap-

[1] `http://yann.lecun.com/exdb/mnist/`

plied entry-wise over the message vector. During training the message is regularized by added noise:

$$\mathrm{DRU}(\hat{m}_i^t) = \sigma(\hat{m}_i^t + \epsilon) \in]0, 1[\tag{2}$$

where σ is the sigmoid function and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ gaussian noise with variance σ_n^2. At test-time the message is discretized:

$$\mathrm{DRU}(\hat{m}_i^t) = \mathbb{I}_{\{\hat{m}_i^t > 0\}} \in \{0, 1\} \tag{3}$$

Letting m_i^t be continuous during training allows for computing gradients through the DRU. The added noise does not impact the gradients and thus the messaging system can be trained. Allowing real-valued messages during training might seem like a naive approach, but the trick here lies in the added noise. If the noise has high enough variance during training the agents can only communicate by selecting very negative or very positive values for $\hat{m}_i^t$. This will result in values of m_i^t close to 0 or 1 respectively. This behavior translates to test-time where the values for m_i^t are in $\{0, 1\}$. The choice of σ_n^2 depends on the environment [Foerster *et al.*, 2016].

It is noteworthy that RIAL and DIAL do not define which agent receives the messages or how to combine messages from many agents. In their experiments Foerster *et al.* [2016] either consider $N = 2$ agents or explicitly define which agent receives each message.

Also DIAL has been evaluated on the previously introduced digit communication game [Foerster *et al.*, 2016]. Agents using DIAL manage to learn an almost perfect messaging protocol with each digit encoded as a 4-bit binary vector.

3.2 Continuous Messages

Communication with continuous messages considers the very general setting of real-valued messages, $\mathcal{M} = \mathbb{R}^L$. Models in this group can easily be designed with completely differentiable messaging mechanisms, avoiding any problems with gradient computations found in discrete message communication.

CommNet

Sukhbaatar *et al.* [2016] have proposed the model Communication Neural Net (CommNet) for multi-agent communication with continuous messages. In the CommNet model agent i selects an action a_i^t and a message $m_i^t \in \mathbb{R}^L$ at timestep t. All messages are then combined by taking the mean

$$\bar{m}^t = \frac{1}{N} \sum_{i=1}^{N} m_i^t \tag{4}$$

and $\bar{m}^t$ is fed to all agents at the next time step. It is possible to only require the agents to act at some timesteps, allowing for multiple rounds of only communication inbetween each action. CommNet is trained using REINFORCE with an RNN for each agent.

A useful property of the CommNet model is that it allows for a dynamic amount of communicating agents throughout a single episode [Sukhbaatar *et al.*, 2016]. This is a direct consequence of the fact that the mean calculation in eq. 4 can be carried out for any amount of input messages N.

Sukhbaatar *et al.* [2016] experiment with the CommNet model in a 4-way traffic junction simulation. Agent cars are assigned a random route to follow and at each time step decide between taking a step along their route or staying in place. Observations are limited to small neighborhoods around each car. If any cars have collided after 40 time steps the episode is considered a failure. The CommNet model achieves a failure rate of 0.3±0.1% (mean and standard deviation over 5 repetitions) in the traffic junction experiment, compared to 15.8±12.5% for independent, non-communicating agents [Sukhbaatar *et al.*, 2016].

IC3Net

Singh *et al.* [2019] consider the task of learning when to communicate. In their setting agents do not just decide what message to send, but also if they should send a message at all or stay silent. To learn this behaviour the messaging part of the CommNet model is extended. The agents select an additional gating action $g_i^t \in \{0, 1\}$ at each time step. The gating action determines if the agent should sent a message or not. Gating is applied to the message output $\hat{m}_i^t \in \mathbb{R}^L$ of the network as $m_i^t = g_i^t \hat{m}_i^t$, before the message is passed to the mean in eq. 4. This extended version of CommNet is named Individualized Controlled Continuous Communication Model (IC3Net). An additional policy $\pi(g_i|\omega_i)$ is introduced with the addition of the gating action. IC3Net is trained using REINFORCE for both the gating and action policies.

TarMAC

A somewhat logical extension to the CommNet model is to consider targeted communication. Das *et al.* [2019] tackle this problem by introducing the Targeted Multi-Agent Communication (TarMAC) model. TarMAC extends CommNet by changing out the mean computation (eq. 4) for an attention mechanism. Attention is a deep learning mechanism popularized by its use in natural language processing [Vaswani *et al.*, 2017; Bahdanau *et al.*, 2014]. The general idea of the mechanism is to allow models to pay more attention to specific parts of available information.

TarMAC implements attention by matching a query from each agent with the incoming messages. The query is a vector q_i^t, output by each agent network at every time step. The messaging part of CommNet is also extended to output both the message m_i^t and a description vector k_i^t. k_i^t can be interpreted as encoding the topic of the message and is matched to the query vectors of other agents. A weighting α_j^{t+1} over incoming messages to agent j at time $t + 1$ is computed as:

$$\alpha_j^{t+1} = \mathrm{sm}\left(\left[\frac{q_j^{t+1\intercal} k_1^t}{\sqrt{d_a}}, \frac{q_j^{t+1\intercal} k_2^t}{\sqrt{d_a}}, \ldots, \frac{q_j^{t+1\intercal} k_N^t}{\sqrt{d_a}}\right]^\intercal\right) \tag{5}$$

where sm is the softmax function and d_a the dimensionality of query and description vectors. The actual message

r_j^{t+1} received by agent j is then:

$$r_j^{t+1} = \sum_{i=1}^{N} \alpha_{ji}^{t+1} m_i^t \quad (6)$$

Since all of the attention mechanism is completely differentiable the model can still be trained with a policy gradient method. Das *et al.* [2019] extend the REINFORCE training used for CommNet to a full Actor-Critic method [Sutton and Barto, 1998]. Following an approach proposed by Lowe *et al.* [2017], the TarMAC model is trained using a centralized critic taking into account the actions of all agents.

Das *et al.* [2019] use a harder version of the traffic junction environment [Sukhbaatar *et al.*, 2016] to evaluate TarMAC. In their experiments the environment contains 4 junctions where cars have to avoid collisions. TarMAC reaches a failure rate of 2.9±1.6%, compared to 21.1±3.4% for just CommNet.

Message Aggregation

Other approaches for combining continuous messages have been proposed in the literature. Peng *et al.* [2017] use a bi-directional RNN to aggregate the messages of agents in their BiCNet model. ATOC [Jiang and Lu, 2018] sets up small communication groups dynamically during episodes and use Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] for intra-group message aggregation.

4 Discussion

Initial models like DIAL and CommNet have shown that multi-agent communication learning is possible and useful. Extensions inspired by the way humans communicate have improved performance on benchmarks created to test the proposed methods. It is clear that this line of research can be useful in applications such as autonomous vehicles or collaborative robotics. Consider for example the traffic junction environment [Sukhbaatar *et al.*, 2016], which easily translates to coordination of autonomous cars in a city.

Most cyber-physical systems in use today feature some form of communication-technology. Deploying communicating agents as parts of such systems could be done for example by wrapping messages between agents in IP-packets. Deploying RL systems can often be hard due to mismatches between the training and deployment environments. With communication learning this is further complicated by possible dropped and delayed messages. These concerns point towards robust communication learning as a useful direction for future research.

Foerster *et al.* [2016] emphasize the possibility to further understanding of how human communication has emerged by studying how communication emerges among machines. Useful results in this direction are however lacking so far. The area is quite young, but it is unclear if the directions being pursued are likely to give any insights into human communication. As in other areas of artificial intelligence research the reliance on neural networks tends to give good results, but offer little new insights to the nature of intelligence.

Many recent works have focused on continuous messages and training by propagating gradients through multiple agents. Eccles *et al.* [2019] on the other hand argue that RIAL is a more naturalistic and applicable method. They point to the fact that RIAL is capable of both decentralized learning and execution. Methods using continuous messages require some centralization for computing gradients during training. It is however not yet clear that decentralized learning is a necessary property in applications. Many machine learning systems are fully trained before being deployed in real-world settings. Updating such systems often include retraining in a protected environment. Decentralized learning is no requirement in such cases. It should still be seen as an important research direction, since there are trends towards erasing the line between training and deployment through online learning methods.

There are other reasons why methods using discrete messages could be of greater interest. Discrete messages are far more interpretable than continuous vectors. This offers greater opportunities to study the resulting communication protocols. Being able to understand the messages would likely be necessary in safety-critical applications where quality properties of systems have to be verified.

This report has focused on communication learning in collaborative settings. Some published work has also touched on semi-cooperative and competitive settings. Experiments have been conducted both with IC3Net and TarMAC in a competitive predator-prey environment [Singh *et al.*, 2019; Das *et al.*, 2019]. The only qualitative conclusion drawn from this is that the prey gains nothing from communicating with the predators hunting it. It would however be unfair to conclude from this that there is no role for communication between competing agents. Learning communication in competitive settings is still a largely unexplored area. Communication in the form of negotiation or threats often occur in adversarial human communication. An interesting direction of future work would be to investigate if similar behavior could be learned through multi-agent RL. To properly test this other competitive settings than predator-prey would have to be designed. Such settings could be used for further experiments with existing models. It would also be of interest to consider communication learning methods with two messaging systems, one for collaborating "teammate"-agents and one for competing "opponent"-agents.

The publications considered in this report were selected based on Google Scholar[2] searches including the keywords *multi agent*, *learning*, *communication*, *communicate* and *reinforcement learning*. The cited publications are believed to be of high scientific quality due to being published in renowned peer-reviewed journals and conferences.

[2] `https://scholar.google.com/`

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[Das *et al.*, 2019] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546, 2019.

[Eccles *et al.*, 2019] Tom Eccles, Yoram Bachrach, Guy Lever, Angeliki Lazaridou, and Thore Graepel. Biases for emergent communication in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 32*, pages 13111–13121. Curran Associates, Inc., 2019.

[Egorov, 2015] Maxim Egorov. Deep reinforcement learning with pomdps. Technical report, Stanford University, 2015.

[Foerster *et al.*, 2016] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc., 2016.

[Foerster *et al.*, 2018] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Jiang and Lu, 2018] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2018.

[Kasai *et al.*, 2008] Tatsuya Kasai, Hiroshi Tenmoto, and Akimoto Kamiya. Learning of communication codes in multi-agent reinforcement learning problem. In *2008 IEEE Conference on Soft Computing in Industrial Applications*, pages 1–6. IEEE, 2008.

[Lin, 1993] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Peng *et al.*, 2017] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games, 2017.

[Singh *et al.*, 2019] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2019.

[Sukhbaatar *et al.*, 2016] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 29*, pages 2244–2252. Curran Associates, Inc., 2016.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.