Learning Communication Protocols with Deep Reinforcement Learning

samuel.r.kopp

October 2019

1 Introduction

Machines understanding and using language is considered an AI-hard problem [Yam13], which I believe is far from solved. With the help of deep learning we can build much better language models, which can be used for example to search for, translate and generate text. These language models are build from training deep neural networks by using giant training sets simply consisting of labelled or unlabelled text data. But I suspect that to build systems that can communicate with humans in a natural way and understand the relationships between words and the real world, we need an approach in which language learning has grounding. An approach of having reinforcement learning agents learn to communicate which each other in order to act better in their environment seems intuitive. However it is not clear what the prerequisites for emergence of human language or other forms of communications in nature are. In this project I want to discover what happens when multiple RL-agents have to share information to fulfill a common objective without a predefined communication protocol. The task in my experiments is inspired by how "simple" animals like bees or ants communicate the location of food sources to each other. The question I want to answer is if neural networks optimized by a state of the art RL algorithm can learn communication in this a task.

2 Methodology

2.1 Multi-Agent Environment

The environment is a $n \times n$ grid in which P patches of food sources are distributed randomly. Each patch of food can be foraged multiple times before it is exhausted. L agents are trying to gather all the food on the grid. for The problem can be formulated as a partially observable Markov decision process (POMDP) in which in every step t each agent i has chooses an action a. This action consists of a message to the other agents $m_t = (m_{1,t}, m_{2,t}...m_{k,t})$ where $m_{j,i} \in V$ with the vocabulary $V = \{1, 2, .., n\}$ and a location (x, y) to explore.

It receives a reward r_i if it lands on a patch of food and a small penalty as a negative reward if not. This reward $R = r_1 + r_2 + ...r_L$ is shared among all agents to motivate cooperation. The actions are chosen by a policy $\pi_\theta : s_i \rightarrow a_i$ where the state s_i consists of the quantity of food and the location of the last exploration by agent i and the messages of all other agents. So each agent only knows about it's own last exploration. The optimal set of messages chosen by the agents would contain something about the quantity and location of the last explored patches, so that the agents can make optimal decisions on which location to choose in the next step.

2.2 Agent Architecture

The policy π_θ is an actor-critic model with a actor network for predicting the next action and a critic network for estimating the expected value of this action. The networks are two-layer MLP with 8 rectified linear units each. The estimated value is used as a baseline reward to reduce variance in the training process.

2.2.1 Single Policy

In this agent architecture a single policy chooses both location and messages to send at each time step. The input state to the policy consists of both all messages and the last observation of agent i. This is perhaps more information as needed for choosing a message to broadcast, since the last observation alone is enough to send a meaningful message to the other agents.

2.2.2 Dual Policy

This architecture consists of a separate policy for sending messages and selecting location to explore. The input state of the exploring-policy is the set of messages sent by all agents in the previous time step. The observation from this exploration is used as the input for the message policy in the same step. The policies do not share any parameters and are separately optimized. While there is a shared reward for the message policy, the exploration policy uses the individual reward of each agent.

2.3 Learning Algorithm

The classical policy gradient objective is $J(\theta) = E(R)$ and its gradient can be computed as $\nabla_\theta J(\theta) = E[\nabla_\theta log(\pi_\theta(a|s))A^\pi(a|s)]$,where the $log(\pi_\theta(a|s))$ is the log-probability of taking action a at state s given policy π_θ and $A^\pi(a|s) = r - V^\pi$ is the estimated advantage of taking this action given the state based on the estimated value V. The optimization is done with the Proximal Policy Optimization(PPO), which optimizes a surrogate loss function for several gradient steps at epoch: $L(s, a, \theta_k, \theta) = min[\frac{\pi_\theta(a|s)}{\pi_k\theta(a|s)}A^\pi(s, a), clip(\frac{\pi_\theta(a|s)}{\pi_k\theta(a|s)}, 1 - \epsilon, 1 + \epsilon)A^\pi(s, a)]$.[Sch+17] PPO achieved state-of-the art results for many game and

control problems, but the choice of optimization method is orthogonal to this work, so the algorithm is not further explained. The value network is optimized with the MSE-loss of the difference of the collected rewards and the estimated values.

2.3.1 Pseudocode of the training loop for single policy

Init buffer, learning rate α and policy π_θ
1.While not converged:
2. reset environment
3. $messages \leftarrow$ list
4. for steps in epoch:
5. reward $\leftarrow 0$
6. for agent in population:
7. $state_i \leftarrow observation_{i,t-1} \cup messages_{t-1}$
8. $m_i, l_i \leftarrow \pi_\theta(state_i)$
9. $messages_t \leftarrow messages_t \cup m_i$
10. $observation_{i,t}, r_i \leftarrow$ environment step l_i)
11. $reward \leftarrow reward \cup r_i$
11. save location, message and state in buffer
12. if all food collected:
13. reset environment
14. save reward in buffer
15. update policy with gradient descent for K steps

2.3.2 Pseudocode of the training loop for dual policy

Init buffer for each policy, learning rate α, policies $\pi_{msg,\theta}$ and $\pi_{loc,\phi}$
1.While not converged:
2. reset environment
3. $messages \leftarrow$ list
4. for steps in epoch:
5. reward $\leftarrow 0$
6. for agent in population:
7. $l_i \leftarrow \pi_{loc,\phi}(messages_{t-1})$
8. $observation_{i,t}, r_i \leftarrow$ environment step(l_i)
9. $m_i \leftarrow \pi_{msg,\theta}(observation_{i,t})$
10. $reward \leftarrow reward \cup r_i$
11. save reward, message and observation in buffer for $\pi_{msg,\theta}$
12. save location and message in buffer for $\pi_{loc,\phi}$
13. if all food collected:
14. reset environment

15. save reward in buffer for $\pi_{loc,\phi}$
16. update both policies with gradient descent for K steps

3 Results

size	population	message-length	average reward	
10	2	3	0.31	
10	10	3	2.52	
5	2	3	10.11	
5	10	3	55.12	single policy results
10	2	0	0.32	
10	10	0	2.55	
5	2	0	10.09	
5	10	0	55.01	

size	population	message-length	average reward	
10	2	3	0.28	
10	10	3	2.52	
5	2	3	10.01	
5	10	3	56.01	dual policy results
10	2	0	0.31	
10	10	0	2.51	
5	2	0	10.22	
5	10	0	55.02	

Clearly there is no significant difference in average reward after the policy optimization converges between the agents that communicate with messages with 3 symbols and the ones that cannot send messages. The result shows that no helpful communication protocol was learned.

4 Related Work

In 2016 OpenAi published a proposal for a language learning paradigm in which agent "understands" language only when it is able to use language productively to accomplish goals in an environment[GM16]. A notable paper by J. Foerster et. al. describes learning a one-bit message protocol by differentiating discrete messages of other agents[Foe+16]. Another recent work from OpenAi shows emergence of a simple abstract compositional language in a multi-agent environment[MA18]. In "On the Pitfalls of Measuring Emergent Communication" researchers from Facebook AI found in their experiments on similar matrix based communication games that their agents do not learn communication.

5 Discussion and Conclusion

It is difficult to figure out what the reason for failing to learn communication in these experiments is. The most obvious explanation is that learning to listen to other agents while other agents have to learn how to pass a message is hard. Reinforcement learning essentially works by enforcing random actions if they give a positive reward. It may be that the required actions for communication occur to rarely to be learned in a stable way. We know that in nature many forms of communication are evolved rather than learned. In some cases we know that bees from different species can learn the dialect of another species.[**6**] Further research could also explore if new individuals in a population can learn communication from already trained agents.

6 References

References

[Yam13] Roman V Yampolskiy. "Turing test as a defining feature of AI-completeness". In: *Artificial intelligence, evolutionary computing and metaheuristics*. Springer, 2013, pp. 3–17.

[Foe+16] Jakob Foerster et al. "Learning to communicate with deep multi-agent reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2137–2145.

[GM16] Jon Gauthier and Igor Mordatch. "A paradigm for situated and goal-driven language learning". In: *arXiv preprint arXiv:1610.03585* (2016).

[Sch+17] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[MA18] Igor Mordatch and Pieter Abbeel. "Emergence of grounded compositional language in multi-agent populations". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.