Learning to learn to communicate

Anonymous Authors[1]

Abstract

How can we teach artificial agents to use human language flexibly to solve problems in a real-world environment? We have one example in nature of agents being able to solve this problem: human babies eventually learn to use human language to solve problems, and they are taught with an adult human-in-the-loop. Unfortunately, current machine learning methods (e.g. from deep reinforcement learning) are too data inefficient to learn language in this way (3). An outstanding goal is finding an algorithm with a suitable 'language learning prior' that allows it to learn human language, while minimizing the number of required human interactions.

In this paper, we propose to learn such a prior in simulation, leveraging the increasing amount of available compute for machine learning experiments (1). We call our approach Learning to Learn to Communicate (L2C). Specifically, in L2C we train a meta-learning agent in simulation to interact with populations of pre-trained agents, each with their own distinct communication protocol. Once the meta-learning agent is able to quickly adapt to each population of agents, it can be deployed in new populations unseen during training, including populations of humans. To show the promise of the L2C framework, we conduct some preliminary experiments in a Lewis signaling game (4), where we show that agents trained with L2C are able to learn a simple form of human language (represented by a hand-coded compositional language) in fewer iterations than randomly initialized agents.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Language is one of the most important aspects of human intelligence; it allows humans to coordinate and share knowledge with each other. We will want artificial agents to understand language as it is a natural means for us to specify their goals.

So how can we train agents to understand language? We adopt the functional view of language (17) that has recently gained popularity (8; 14): agents understand language when they can use language to carry out tasks in the real world. One approach to training agents that can use language in their environment is via *emergent communication*, where researchers train randomly initialized agents to solve tasks requiring communication (7; 16). An open question in emergent communication is how the resulting communication protocols can be transferred to learning human language. Existing approaches attempt to do this using auxiliary tasks, for example having agents predict the label of an image in English while simultaneously playing an image-based referential game (12). While this works for learning the names of objects, it's unclear if simply using an auxiliary loss will scale to learning the English names of complex concepts, or learning to use English to interact in an grounded environment.

One approach that we know will work (eventually) for training language learning agents is using a human-in-the-loop, as this is how human babies acquire language. In other words, if we had a good enough model architecture and learning algorithm, the human-in-the-loop approach . However, recent work in this direction has concluded that current algorithms are too sample inefficient to effectively learn a language with compositional properties from humans (3). Human guidance is expensive, and thus we would want such an algorithm to be as sample efficient as possible. An open problem is thus to create an algorithm or training procedure that results in increased sample-efficiency for language learning with a human-in-the-loop.

In this paper, we present the Learning to Learn to Communicate (L2C) framework, with the goal of training agents to quickly learn new (human) languages. The core idea behind L2C is to leverage the increasing amount of available compute for machine learning experiments (1) to learn a 'language learning prior' by training agents via meta-learning in

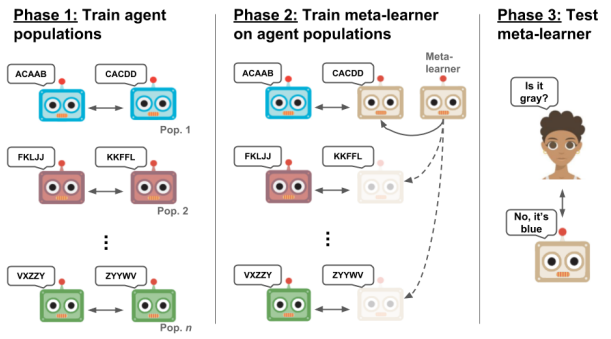

Figure 1. Diagram of the L2C framework. An advantage of L2C is that agents can be trained in an external environment (which *grounds* the language), where agents interact with the environment via actions and language. Thus, (in theory) L2C could be scaled to learn complicated grounded tasks involving language.

simulation. Specifically, we train a meta-learning agent in simulation to interact with populations of pre-trained agents, each with their own distinct communication protocol. Once the meta-learning agent is able to quickly adapt to each population of agents, it can be deployed in new populations unseen during training, including populations of humans. The L2C framework has two main advantages: (1) permits for agents to learn language that is *grounded* in an environment with which the agents can *interact* (i.e. it is not limited to referential games); and (2) in contrast with work from the instruction following literature (2), agents can be trained via L2C to both *speak* (output language to help accomplish their goal) and *listen* (map from the language to a goal or sequence of actions).

To show the promise of the L2C framework, we provide some preliminary experiments in a Lewis signaling game (4). Specifically, we show that agents trained with L2C are able to learn a simple form of human language (represented by a hand-coded compositional language) in fewer iterations than randomly initialized agents. These preliminary results suggest that L2C is a promising framework for training agents to learn human language from few human interactions.

2. Learning to learn to communicate

L2C is a training procedure that is composed of three main phases: (1) **Training agent populations:** Training populations of agents to solve some task (or set of tasks) in an environment. (2) **Train meta-learner on agent populations:** We train a meta-learning agent to 'perform well' (i.e. achieve a high reward) on tasks in each of the training populations, after a small number of updates. (3) **Testing the meta-learner:** testing the meta-learning agent's ability to learn new languages, which could be both artificial (emerged languages unseen during training) or human.

A diagram giving an overview of the L2C framework is shown in Figure 1. Phase 1 can be achieved in any number of ways, either through supervised learning (using stochastic gradients) or via reinforcement learning (RL). Phases 2 and 3 follow the typical meta-learning set-up: to conserve space, we do not replicate a formal description of the meta-learning framework, but we direct interested readers to Section 2.1 of (6). In our case, each 'task' involves a separate population of agents with its own emergent language. While meta-training can also be performed via supervised learning or RL, Phase 3 must be done using RL, as it involves interacting with humans which cannot be differentiated through. See Section 5 for a discussion of each of these phases in more detail.

3. Problem set-up

A speaker-listener game We construct a referential game similar to the Task & Talk game from (11), except with a single turn. The game is cooperative and consists of 2 agents, a speaker and a listener. The speaker agent observes an object with a certain set of properties, and must describe the object to the listener using a sequence of words (represented by one-hot vectors). The listener then attempts to reconstruct the object. More specifically, the input space consists of p *properties* (e.g. shape, color) and t *types* per property (e.g. triangle, square). The speaker observes a symbolic representation of the input x, consisting of the concatenation of p one-hot vectors, each of length t. The number of possible inputs scales as t^p. We define the vocabulary size (length of each one-hot vector sent from the speaker) as $|V|$, and fix the number of words sent to be w.

Approximating human language To simulate a simplified form of human language on this task, we programatically generate a perfectly compositional language, by assigning each 'concept' (each type of each property) a unique symbol. In other words, to describe a blue shaded triangle, we create a language where the output description would be "blue, triangle, shaded", in some arbitrary order and without prepositions. By 'unique symbol', we mean that no two concepts are assigned the same word. By generating this language programmatically, we avoid the need to have humans in the loop for testing, which allows us to iterate much more quickly. This is feasible because of the simplicity of our speaker-listener environment; we do not expect that generating these programmatic languages is practical when scaling to more complex environments. We call these agents compositional bots (CBs).

Implementation details The message produced by the speaker is a sequence of p categorical random variables

Game size	Avg. train %	Avg. test %
$p = 3, t = 5$	44.2	2.5
$p = 6, t = 10$	98.5	97.3

Table 1. Results from training speaker-listener pairs on the referential game, using emergent communication with categorical random variables (cat-TPs). Performance is measured in terms of % accuracy in reconstructing the correct input.

which are discretized using Gumbel-Softmax with an initial temperature $\tau = 1$. We set the vocabulary size to be equal to the total number of concepts $p \cdot t$. In our initial experiments, we train our agents using Cross Entropy which is summed over each property p. We use the Adam optimizer (10) with a learning rate of 0.001. We first demonstrate the results with a meta-learning listener (a *meta-listener*), that learns from the different speakers of each training population.

In Section 4 below, we perform L2C experiments varying the type and number of training populations, speaker and listener parameterizations, and listener meta-learning algorithms. Unless otherwise specified, we use an over-parameterized linear policy (i.e. an MLP with linear activations and 500 'hidden units') for the speaker and listener, and the meta-listener is trained with a first-order version of MAML (6) on 50 purely compositional training populations (CBs, see Section 4.1).

4. Experiments

4.1. Obtaining agent populations

We first obtain encoder-decoder pairs that solve the game described in Section 3, which we can use for our meta-learning procedure.

Random compositional languages Using the procedure described in Section 3, we can vary the random seed to obtain a huge number of CBs. Meta-training on CBs is a good benchmark for whether L2C can help agents learn compositional structure.

Emergent communication We can also train agents to solve the task with emergent communication, and use them for meta-learning; we call these agents 'trained populations' (TPs). We train two variants of TPs, both using Gumbel-Softmax: in one, the speaker's words are represented by a sequence of p categorical random variables (cat-TPs), each of size $p \cdot t$. Thus, each variable is a one-hot vector representing a word. We also train speakers to output a sequence of $p \cdot \lceil \log_2(t) \rceil$ binary random variables (bin-TPs), which we take to represent a binary encoding of one-hot vectors.

We show the results of training cat-TPs in Table 1 for two different game sizes: a smaller game similar to (11) with $p = 3, t = 5$ (125 possible inputs), and a larger game with $p = 6, t = 10$ (10^6 possible inputs). We find that in both cases cat-TPs are able to achieve a low error on the training set when passing discrete vectors ($\tau = 0$), however the agents only generalize to the test set in the larger game. Thus, in our remaining L2C experiments we focus on the more interesting case with $p = 6, t = 10$.

Interestingly, while both cat-TPs and bin-TPs generalize to the test set in the $p = 6, t = 10$ case, bin-TPs are perfectly compositional (changing one property of the input changes only one of the speaker's words), whereas cat-TPs are not. Thus, bin-TPs are equivalent to CBs. We perform experiments below with both cat-TPs and CBs.

4.2. Training the meta-learner

Here, we describe our initial results into the factors affecting the performance of L2C. Since our ultimate goal is to transfer to learning a human language in as few human interactions as possible, we measure success based on the number of samples required for the meta-learner to reach a certain performance level (95%) on a held-out test population, and we permit ourselves as much computation during pre-training as necessary.

Varying meta-learning algorithms We experimented with several algorithms to train our meta-listener: a randomly initialized agent, an agent pre-trained on all populations that performs $n = 1$ update per population, the Reptile algorithm (15) that performs $n > 1$ updates per population, and a first-order variant of MAML (FOMAML) (6). In our initial experiments, we found that the Reptile and the pre-training agent didn't improve significantly over the random initialization baseline. However, when we have enough training populations, the FOMAML algorithm improved significantly over the randomly initialized baseline, and even approaches the minimum number of examples a human would need to solve the task (60, one for each word in the vocabulary) — see Figure 2.

Varying listener parameterizations We tried various models for the meta-listener, including an LSTM, an MLP, and a linear model. While L2C with FOMAML worked in all of these cases, we found the best results with an over-parameterized linear model (a 1-layer MLP with linear activations). Due to performance and speed, we thus used the linear model in the remainder of our experiments. We suspect the over-parameterization helped (over a regular linear model) due to improved gradient descent dynamics.

Varying the type of training populations One of the core questions around L2C is what kind of training populations are required in order for the meta-learner to gen-

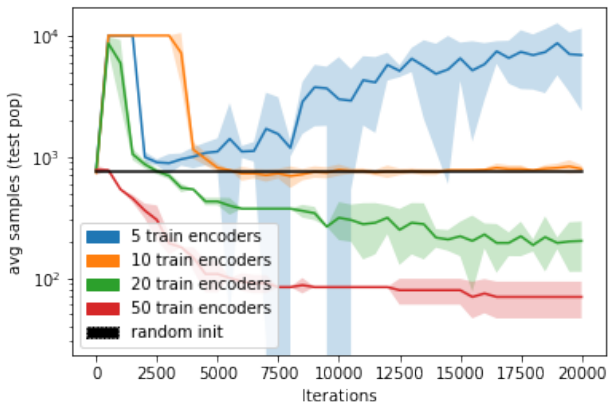

Figure 2. Performance of the meta-learner (in terms of number of training samples required to achieve >95% test accuracy on a test population) over the course of meta-training (horizontal axis), while varying the number of training encoders. Results averaged over 5 seeds, with standard error shown.

eralize to human language. Here we experiment with both non-compositional cat-TPs and compositional CBs (equivalent to bin-TPs). We find that agents trained with L2C on cat-TPs do not show improved generalization performance compared to random initialization, whereas agents trained on CBs do (Figure 2). Thus, having compositional training populations is necessary for agents to learn compositional structure to generalize to compositional test populations.

Varying the number of training populations As can be inferred from Figure 2, having more training populations improves performance. Having too few training populations (eg: 5 train encoders) results in overfitting to the set of training populations and as the meta-learning progresses, the model performs worse on the test populations. For more than 10 training encoders, models trained with L2C require fewer samples to generalize to a held-out test population than a model not trained with L2C.

5. Outstanding challenges

There are several immediate directions for future work: training the meta-agent via RL rather than supervised learning, and training the meta-agent as a joint speaker-listener (i.e. taking turns speaking and listening), as opposed to only listening. We also want to scale L2C training to more complicated tasks involving grounded language learning, such as the Talk the Walk dataset (5), which involves two agents learning to navigate New York using language.

More broadly, there are still many challenges that remain for the L2C framework. In fact, there are unique problems that face each of the phases described in Section 2. In Phase 1, how do we know we can train agents to solve the tasks we want? Recent work has shown that learning emergent communication protocols is very difficult, even for simple tasks (13). This is particularly true in the multi-agent reinforcement learning (RL) setting, where deciding on a communication protocol is difficult due to the non-stationary and high variance of the gradients (13). This could be addressed in at least two ways: (1) by assuming the environment is differentiable, and backpropagating gradients using a stochastic differentiation procedure (9; 14), or (2) by 'seeding' each population with a small number of human demonstrations. Point (1) is feasible because we are training in simulation, and we have control over how we build that simulation — in short, it doesn't really matter *how* we get our trained agent populations, so long as they are useful for the meta-learner in Phase 2.

In Phase 2, the most pertinent question is: how can we be sure that a small number of updates is sufficient for a meta-agent to learn a language it has never seen before? The short answer is that it doesn't need to completely learn the language in only a few updates; rather it just needs to perform better on the language-task in the host population after a few updates, in order to provide a useful training signal to the meta-learner. For instance, it has been shown that the model agnostic meta-learning (MAML) algorithm can perform well when multiple gradient steps are taken at test time, even if it is only trained with a single inner gradient step. Another way to improve the meta-learner performance is to provide a *dataset* of agent interactions for each population. In other words, rather than needing to meta-learner perform well after interacting with a population a few times, we'd like it to perform well after doing some supervised learning on this dataset of language, and after a few interactions. After all, we do have lots of available datasets of human language, and not using this seems like a waste.

Finally, in Phase 3, how do we know that the meta-learner will be able to generalize to learn a human language? This comes down to the similarity between the training populations and human language, and the diversity of the training populations. For instance, we expect certain properties like compositionality to be important in the training populations for transferring to human languages, which are inherently compositional. But the training languages may not need to be very close to human language; as a comparison, significant progress has been made in transferring robots from simulation the real world using *domain randomization*, i.e. by adding random image textures during training. Even though these image textures look nothing like the real world, it helps improve robustness of the learner, which allows it to generalize. Providing a detailed examination of the required properties of the trained population languages, such that a meta-learner is able to generalize to human language, is an important direction for future work.

Acknowledgements We'd like to thank Jean Harb, Liam Fedus, Cinjon Resnick, Amy Zhang, Igor Mordatch, and others at MILA and Facebook AI Research for discussions related to the ideas in this paper. RL is funded in part by a Vanier Scholarship.

References

[1] D. Amodei and D. Hernandez. Ai and compute. *https://blog. openai. com/aiand-compute*, 2018.

[2] D. Bahdanau, F. Hill, J. Leike, E. Hughes, P. Kohli, and E. Grefenstette. Learning to follow language instructions with adversarial reward induction. *arXiv preprint arXiv:1806.01946*, 2018.

[3] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: First steps towards grounded language learning with a human in the loop. *arXiv preprint arXiv:1810.08272*, 2018.

[4] L. David. Convention: a philosophical study, 1969.

[5] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

[6] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[7] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *CoRR*, abs/1605.06676, 2016.

[8] J. Gauthier and I. Mordatch. A paradigm for situated and goal-driven language learning. *arXiv preprint arXiv:1610.03585*, 2016.

[9] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[11] S. Kottur, J. M. Moura, S. Lee, and D. Batra. Natural language does not emerge'naturally'in multi-agent dialog. *arXiv preprint arXiv:1706.08502*, 2017.

[12] A. Lazaridou, A. Peysakhovich, and M. Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[13] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[14] I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[15] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR, abs/1803.02999*, 2018.

[16] O. Tieleman, A. Lazaridou, S. Mourad, C. Blundell, and D. Precup. Shaping representations through communication. *OpenReview*, 2018.

[17] L. Wittgenstein. Philosophical investigations. 1953.