Protecting Communications with Reinforcement Learning in a Multi-Agent Game

Faris Sbahi[1] and Barrett Ames[2]

[1] Primary
faris.sbahi@duke.edu
[2] Consultant
barrett.ames@duke.edu

Abstract. Here we explore the ability of reinforcement learning (RL) agents to cooperate to establish a secure communication protocol in the face of an adversary without explicit communication. Hence, we construct a general sum mixed cooperative-competitive game to study the effectiveness of various multi-agent RL algorithms to perform in this environment. We compare our results with parallel experiments involving generative adversarial networks. We conclude that the Multi-Agent Deep Deterministic Policy Gradient algorithm prescribed in [1] converges toward the desired behavior of securing protected communications in a period of time with which the other RL algorithms do not, and furthermore it represents a stronger result than the one involving GANs.

Keywords: reinforcement learning, multi-agent, communication, cryptography, generative adversarial networks, multi-agent actor-critic

1 Introduction

Reinforcement learning (RL) has been applied to increasingly complex tasks, ranging from Atari games to robotics. In most cases, the single-agent paradigm is pervasive given that predicting the behavior of other actors is unneeded.

Nevertheless, studying multi-agent systems is increasingly important if we expect to scale RL to interactive systems rather than applying the technique in a vacuum. We may expect that some sort of emergent, co-evolutionary behavior would evolve in a multi-agent setting. Hence, we can ask questions pertaining to social dilemmas [2] and language [3], reaching Nash equilbria in cooperative or competitive games [4], or the ability to develop communication to achieve cooperative control tasks [5] or even solve riddles [3].

Here we consider a mixed cooperative-competitive where an agent must communicate to another agent in the presence of an adversary. We find this worth considering because it allows us to study (1) the effectiveness of various RL and non-RL algorithms in a Markov game (defined below) of this type, (2) whether agents can protect their communications without being provided with an explicit cryptographic algorithm, and more broadly (3) whether and/or what kind of equilibria we observe depending on our algorithm and reward structures.

Ultimately, we show that the simple multi-agent extensions of popular reinforcement learning algorithms perform poorly while the Multi-Agent Deep Deterministic Policy Gradient algorithm developed in [1] produces a competitive game. Furthermore, the GAN context described in [6] seems to allow for protected communications between agents but we'll argue that it's likely because of the limited decryption capability one might expect in this setup.

1.1 The Problem

The general game is simple.

We seek to define 3 agents Alice, Bob, and Eve such that Alice communicates a message to Bob who must reconstruct the message. Furthermore, adversary Eve observes the channel and additionally seeks to reconstruct the message. Hence, Alice and Bob should be penalized based off of Eve's reconstruction and rewarded based off of Bob's. Furthermore, Alice and Bob use a symmetric, randomly generated key (not available to Eve) to encode their message in a manner with which they define.

2 Methods

2.1 Markov Games

First, we must extend the traditional Markov Decision Process (MDP) to the multi-agent setting. This is well documented and termed as a Markov Game [2]. An N-agent Markov game is given by a collection states $\{S\}$ with which any agent can take, in addition to actions $\{A_1 \cdots A_N\}$ and observations $\{O_1 \cdots O_N\}$. The ith agent uses a stochastic policy $\pi_i : O_i \times A_i \to [0, 1]$ which takes the agent to the next state according to the state transition function $\mathcal{T} : S \times A_1 \times \cdots \times A_N \to S$. Similarly, the ith agent obtains rewards as a function of the state and action, and receives an observation. Each agent seeks to maximize its total expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ where $r_i : S \times A_i \to \mathbb{R}$ is the reward associated with the taken action with γ as the discount factor and T the time horizon.

2.2 Extending Popular Single-Agent Algorithms

With this notation in mind, we can readily popular single-agent algorithms to the multi-agent paradigm.

Q-learning uses an action-value function for a policy which can be written recursively as $Q^\pi(s, a) = \mathbb{E}_{s'}[r(s, a) + \gamma \mathbb{E}_{a'}[Q^\pi(s', a')]]$. Hence, Q-learning can be extended by simply allowing each agent to learn its independently optimal Q_i.

Policy gradient can likewise be extended for example by approximating by again having each agent learn its own Q_i.

2.3 Multi-Agent Deep Deterministic Policy Gradient

Multi-Agent Deep Deterministic Policy Gradient is described by Lowe et. al [1] and is noteworthy because it adopts centralized training with decentralized execution. Hence, it is an actor-critic policy gradient algorithm where the centralized critic is augmented with information about the policies of other agents. The motivation is that if the actions taken by all agents are known, then environment is stationary even as the policies change. So, the centralized Q action-value function takes in the inputs of all agents and outputs the Q-value for each agent.

3 Theoretical Matters

There are clear issues with the popular single-agent extensions.

First, in the case of Q-learning, when each agent updates its individual Q_i the environment appears non-stationary hence violating the Markov property. Furthermore, experience replay in the case of DQN is no longer useful because of each agent's policy being dependent on the policy implemented by the other agents.

Furthermore, policy gradient methods are known to be of high-variance and this is exacerbated in multi-agent settings [1].

Conceptually, it's clear why MADDPG is particularly useful in our case. If training is centralized and the policies are used to train each agent in our cryptography game, then we can expect added competition and the ability for each agent to uncover motivations in its competitor.

4 Experiments

4.1 GANs

Following the experimental setup defined in [6], we setup a mirror experiment with code written in TensorFlow to observe whether neural networks can learn to protect communications between one another.

In summary, we had Alice, Bob, and Eve represented as neural networks. Alice has inputs P and K corresponding to the plaintext message and symmetric key and output C the ciphertext. Bob has inputs C and K with output P_B which is Bob's decryption of P. Finally, Eve simply has input C and output P_E.

Eve has the simple loss function which corresponds to the L1 distance between P_E and P. Alice and Bob share the loss function which corresponds to L1 distance between P_B and P minus $(N/2 - \text{Eve L1 error})^2/(N/2)^2$ where N is the length of P. The authors note the peculiarity of this loss function.

In addition to this peculiarity, the authors concede that they had to cut a number of corners to achieve the expected behavior. Note also that the agents are trained in an alternating pattern, switch back and forth between training Eve and then Alice and Bob together. Each epoch represents having trained both parties once.

In our experiments we let the size of the plaintext and key be 4 and the range of values each of the four entries can be -1 to 1. As in Figure 1, we found that the adversary quickly loses ability to decrypt and Alice and Bob converge to an effective protocol. Nevertheless, in all of our experimentations we found that once Eve drifted away from being able to decrypt the message, it never erred back away from randomness. Hence, this may represent a weakness of adversaries in this setup.

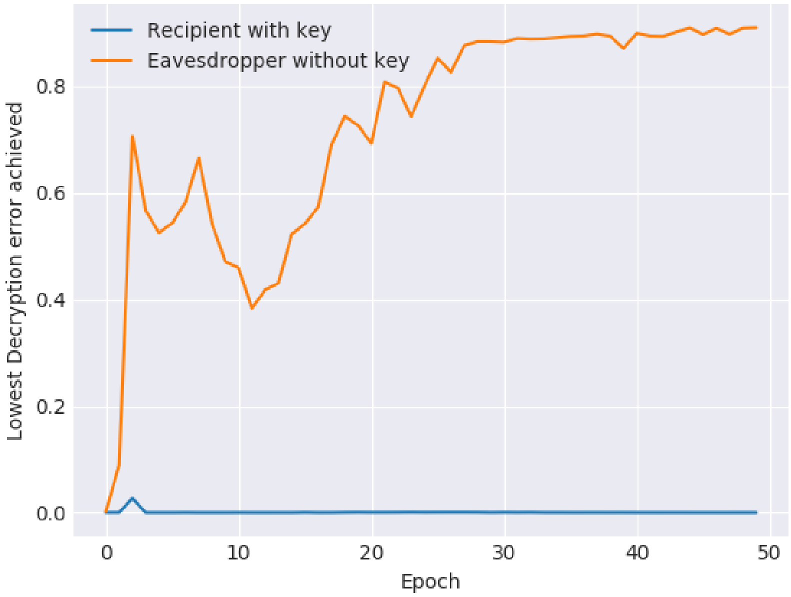

Fig. 1: Results observed using GANs

4.2 DQN and Policy Gradient

We omit any charts because we failed to have either algorithm converge to expected behavior in our experimentation given the number of episodes that time permitted i.e. Adam and Bob could not communicate without Eve fully decrypting the message.

Nevertheless, this fits well with our theoretical considerations. We used an open source multi-agent reinforcement learning testbed (`https://github.com/sisl/MADRL`) to gather these results.

4.3 Multi-Agent Deep Deterministic Policy Gradient

Here we used OpenAI Gym and OpenAI multi-agent-environment-specific code to define our environment and cryptography game. Furthermore, we use TensorFlow to develop the MADDPG algorithm. We validated the algorithm by testing it on a "predator-prey" environment open sourced by OpenAI. We trained the algorithm for 1000 episodes with 1000 steps per episode on a GPU-compute optimized instance on AWS.

We defined our Markov game as analogously as possible to above. We allowed for size N symmetric keys and plaintexts with N values ranging from -1 to 1. Hence, we uniformly at random select initial values for each of these in $[-1, 1]$. Furthermore, we allow a continuous action space across this range for each agent; we define an action to be a length N message that is broadcasted to all agents. Hence, ideally, Alice and Bob can arrange some coded manner of speaking with one another that uses the hidden information (key) and hence won't be learned by Eve. We choose $N = 4$ to be consistent. Alice and Bob are rewarded for minimizing the difference between Bob's interpretation of Alice's broadcast message and the original messaged and penalized for Eve's correct interpretation of the message. Eve is rewarded for its correct interpretation of the message.

Now, applying the MADDPG algorithm described above, we noticed that after a few hundred episodes Adam and Bob would pull away from the adversary. Nevertheless, it remains a competitive game and Eve often is able to recover. This may represent Alice and Bob creating a simple coded-language policy which doesn't combine the key and hence can be learned by Eve.

Ultimately, Adam and Bob generally are able to reach an equilibrium and pull away entirely after about 500 episodes. This may represent Alice and Bob determining an effective policy combining the key.

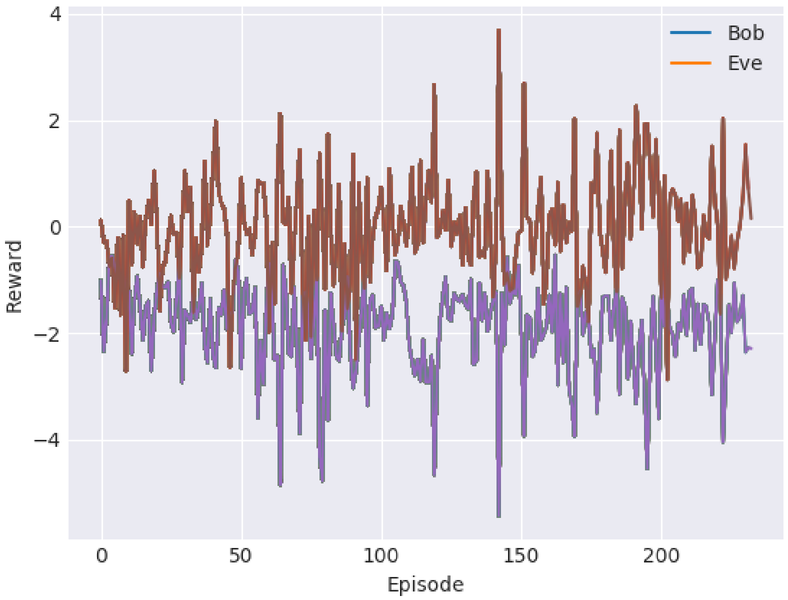

Fig. 2: Results observed using MADDPG. Bob (Brown) and Eve (Purple).

An example is shown in Figure 2. Here we see a slight pulling away after 150 episodes, only to be met by Eve's full recovery and then more pulling away. In our experiments, we found an equilibrium of secure communication after about 500 episodes.

4.4 Brief Comparisons

It's unfair to make a direct comparison between the GAN and MADDPG set-ups because the games have a technically different fundamental structure. Nevertheless, we can still define our goal as to secure some means of private communication between agents in the face of *the best possible* adversary. Hence, as we noted above, Eve seemed particularly weak and unable to recover from large enough perturbations from accurate decryption in the GAN case. Furthermore, our goal of keeping the emergence of a protected communication protocol as "unsupervised" as possible is clearly best achieved by the RL algorithms because it did not require the "cutting of corners" described by the authors of the GAN paper.

Finally, one can at least note that the MADDPG algorithm converged at a speed with which the other algorithms could not. This fits with our theoretical considerations.

5 Conclusion

We showed that it is difficult to derive meaningful results when using the classic RL algorithms in the multi-agent setting theoretically and empirically in the lens of protected communications.

Nevertheless, interesting results are feasible with GANs though they might be weaker in nature than the results found using MADDPG given the weak ability of the adversary in the GAN context.

5.1 Further Development

Other interesting multi-agent reinforcement learning algorithms exist and would be worth testing [5]. Furthermore, we redefined the precise structure of our game several times during development, varying what types of actions each agent can make in addition to the reward structure. If time had allowed, it would've been useful to quantify the effect of these changes. It's also interesting to consider the case of having a greater number of adversaries or "good" communicators.

We'd also like to analyze the convergent policy in the RL setting and determine if it's some clear kind of combination with the shared symmetric key.

Finally, it could be interesting to mix GANs with RL agents. Perhaps, then, we can show that a MADDPG adversary is a better approximation of the best possible adversary.

5.2 Thanks

Thanks to Prof. Parr for giving me enough time to jam this project into a short period of time. I was able to learn how to run deep reinforcement learning experiments in the cloud, understand a bit about cryptography and GANs, develop an understanding of Markov games, multi-agent reinforcement learning, and some interesting algorithms that have been developed to fill the holes that I mentioned above in the simple single-agent extension algorithms.

Thanks to Barrett for consulting on this project and being flexible with his schedule in addition to being happy to provide feedback.

References

1. R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

2. J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, “Multi-agent reinforcement learning in sequential social dilemmas,” in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 464–473, International Foundation for Autonomous Agents and Multiagent Systems, 2017.
3. J. Foerster, Y. Assael, N. de Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” in *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.
4. L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, “Robust adversarial reinforcement learning,” *arXiv preprint arXiv:1703.02702*, 2017.
5. J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83, Springer, 2017.
6. M. Abadi and D. G. Andersen, “Learning to protect communications with adversarial neural cryptography,” *arXiv preprint arXiv:1610.06918*, 2016.
7. R. Elderman, L. J. Pater, A. S. Thie, M. M. Drugan, and M. Wiering, “Adversarial reinforcement learning in a cyber security simulation.,” in *ICAART (2)*, pp. 559–566, 2017.
8. S. Shiva, S. Roy, and D. Dasgupta, “Game theory for cyber security,” in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, p. 34, ACM, 2010.
9. C. Finn, P. Christiano, P. Abbeel, and S. Levine, “A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models,” *arXiv preprint arXiv:1611.03852*, 2016.
10. D. Pfau and O. Vinyals, “Connecting generative adversarial networks and actor-critic methods,” *arXiv preprint arXiv:1610.01945*, 2016.
11. X. Lin, P. A. Beling, and R. Cogill, “Multi-agent inverse reinforcement learning for zero-sum games,” *arXiv preprint arXiv:1403.6508*, 2014.
12. R. Sutton and A. Barto, “Reinforcement learning: an introduction.[internet],” 2016.