Project Proposal: Collaborative Policy Learning for Vehicle Navigation

Chris Ying
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
cying@andrew.cmu.edu

Ananya Kumar
School of Computer Science
Carnegie Mellon University
ananyak@andrew.cmu.edu

Dominick DiRenzo
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
ddirenzo@andrew.cmu.edu

1 Overview

Many recent advances in reinforcement learning have focused on single-agent (e.g. Atari [2]) or competitive (e.g. Go [3]) environments. We will explore the task of learning a single policy that controls multiple agents collaborating to complete a task. For a real-world example, consider autonomous vehicles: each vehicle will be running the same code so they all share the same policy and have to "collaborate" to avoid collisions while also getting to the destination.

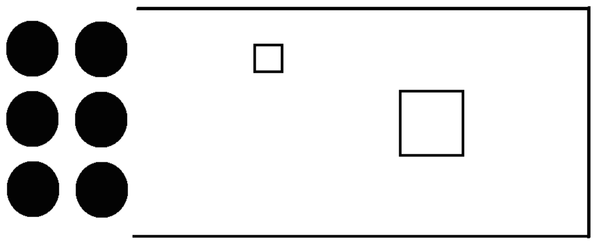

Figure 1: Cars are the shaded black circles, end of lane is on the right, rectangles are objects

Environment We simulate a group of cars trying to drive across a road lane with a randomly generated obstacle (see figure 1). Let c_i be the time it takes car i to reach the end of the road, the goal is to minimize the sum of c_i. The cars are identical and circular. The time steps are small discrete steps. Each time step, each car can accelerate in any direction (within imposed limits). Importantly, all the cars must act under the same policy. Each car will only be able to see a local rectangular region of the map. A car's state consists of frames of the local area for the last n steps, and the current velocity and direction. We will have a negative reward for collisions between cars and a positive reward for reaching the destination quickly.

Learning We will use a policy gradient approach. We will set up a neural net that takes in the last few frames of the local map, the current velocity and direction. Based on this input, the neural net needs to output a distribution over possible actions. We will use a Gaussian parametrization, that is, the neural net will output μ_v, σ_d, which are the mean and standard deviation of a Gaussian distribution, and we will sample the velocity from this distribution, and similarly for the car direction (mod 2π). The neural net architecture will involve a CNN, which takes in the local map frames. This feeds into a few fully connected layers, which takes the CNN, and the current velocity and direction of the car, as input. To train this policy network, we will use a policy gradient approach, such as REINFORCE.

Testing For now, we will fix the size of the lane, and add a single fixed size obstacle at a random location. There will always be a way for all the cars to get to the end of the road. When training, we will generate a random map based on this specification, and run an episode, applying REINFORCE to train. To measure test performance, we will save our final policy, generate random maps, and compute the average performance on the generated maps.

2 Literature Review

We plan on using the policy gradient methods discussed in class. For this we will use the course lecture notes as well as Sutton and Barto's book[4]. We will be dealing with continuous action spaces so section 13.7 will be useful in particular.

There has been some recent work on multi-agent cooperation, however most of it focuses on 2 agents. We may be able to extend the ideas to a larger number of agents. A recent paper from DeepMind[1] applies reinforcement learning to cooperative 2 player games involving fruit gathering and hunting in a wolfpack. Another paper from Tampuu, Matiisen, et al[5] studies cooperative 2-player pong, where they attempt to make a game last as long as possible.

Ming Tan's old but highly cited paper[6] on cooperative learning outlines methodology and results that we can apply to our project.

3 Project Plan

By the second milestone we hope to have:

- The simulation environment written in OpenAI Gym.
- A basic convolutional network set up in Tensorflow for learning the policy function.
- A framework to train the policy network via policy gradients.

By the end of the project we hope to have:

- A fine-tuned policy network for the task.
- A decent training framework that primarily uses policy gradients but may use some supervised learning if necessary for early training.
- Some high-level theoretical explanations of any phenomenon we may observe.

4 Computational Resources

The most computationally expensive part of our project is the evaluation of the convolutional policy network. Ideally we would like to have a decent GPU (K20 or better) for maybe 1 week of total computation time (conservative estimate).

References

[1] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[4] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[5] 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.

[6] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.