Sean Whalen, Evan McKee, Cody Rooks
Final Project

ECE 517 Fall 2018
Due 11/16/2018

Multi-Agent Reinforcement Learning

GENERAL IDEA

We would like to investigate multi-agent reinforcement learning (MARL) for cooperative and competitive-cooperative problems. In a fully cooperative scenario all agents have the same agenda, while in a competitive-cooperative construction there may be conflicting as well as common goals.

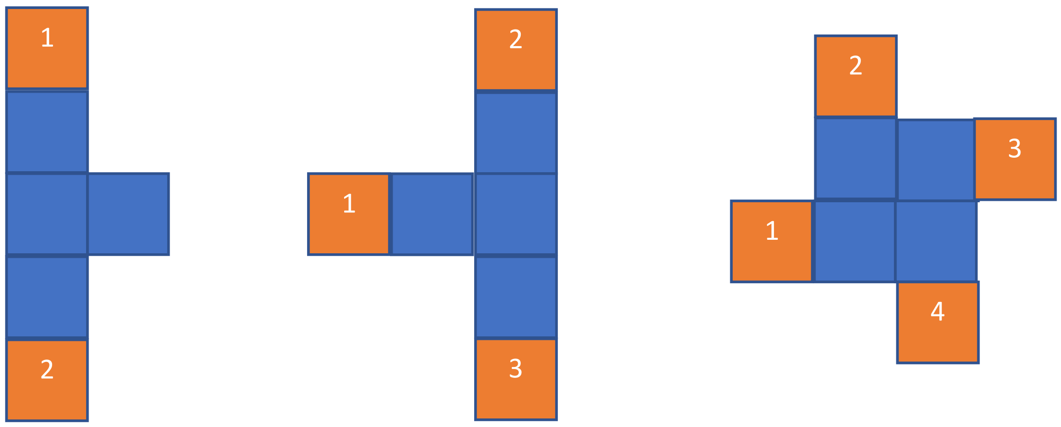
PAST WORK AND NOVELTY

While there is an abundance of research for MARL, there are 2 papers that will mainly use to guide our project. One considers competitive agents without communication and one that focuses on fully cooperative scenarios with communication. They are both well cited, and the latter includes a github repository that we will use for reference if we find ourselves needing to use PyTorch.

We consider our experiment useful and novel, because it will compare learning rates against an optimal agent found with DP and will experiment with communication in a competitive-cooperative environment. The algorithms learned in class, including MC, Sarsa, and Q-learning, will be applied and expanded to accommodate multi-agent problems.

THE EXPERIMENT

To keep things simple and grid based, we will deploy our MARL algorithms on a few different constructions of a Pinwheel game. Each agent is labeled with a number in an orange box and desires to reach the location of one of the other agents' starting places. Collision is inevitable, and each agent will learn to make decisions based on the other agents' policies. The board can be made larger to help visualize the learning taking place.



A 2, 3, and 4 agent construction of the Pinwheel game

To provide context for our MARL we'll first use DP with one agent that controls all actors and will be rewarded only when every orange box reaches its destination. We will use a discount to encourage policies that solve the puzzle in fewer steps. This process will also give us insight as to whether we can use a tabular approach. If the state space is too large we will switch to function approximation, using the simplest neural network as we can, and use semi-gradient Q-Learning.

Then we will transition to the separating the learning, making each orange box an agent with an individualized policy and action value function. The location of the other agents now becomes part of one agents state. In this case each agent does not know how the others will move at each time step, so it is essentially model-free. We will start with a Monte Carlo approach with Sarsa, Expected Sarsa or Q-Learning. Again, if the state space is too large, we will instead a simple neural network that backpropagates TD-Error or error with respect to some other target. Since this is a non-stationary problem, we will not use replay. At first, we will keep the same reward structure as in the master-agent case, and then compare with an approach that rewards each agent individually for finding its respective destination (the cooperative-competitive scenario). This will be an episodic task that ends when every orange box finds its destination or when a max iteration count has been exceeded.

We fully expect that the multi-agent approach cannot exceed the performance of the master-agent for the cooperative scenario, because the master-agent has full knowledge whereas the MARL does not. Considering this, we'd like to measure the impact of communication. If all the agents share their intent for the next move before acting, can better performance be achieved? What about if the agents know where the other ones want to go? This is the interesting part of the project, where we will experiment with communication protocols and their effects. We also may introduce a history aspect, to see if that helps agents guess the intentions of others.

If we can operate within a tabular environment, we are confident we can quickly write all the algorithms / code we need ourselves in python3 with numpy. We aren't familiar with practical coding of deep learning / neural networks so if that becomes necessary, we will attempt to install and use PyTorch.

CONCERNS

We are skeptical that our agents will learn sequences of moves rather than concepts like "2 wants to go behind me, so I should move out of the way." Perhaps a neural network approach and randomizing the environment between episodes as well as agent goal squares could be future work.

CITATIONS

Multi-agent Reinforcement Learning in Sequential Social Dilemmas
https://dl.acm.org/citation.cfm?id=3091194

Learning to Communicate with Deep Multi-Agent Reinforcement Learning
http://papers.nips.cc/paper/6042-learning-to-communicate-with-deep-multi-agent-reinforcement-learning.pdf

Sean Whalen, Evan McKee, Cody Rooks — ECE 517 Fall 2018

Final Project — Due 11/16/2018

MEETING MINUTES

Attendees:

- Sean
- Evan
- Cody
- Dr. S

- justify why the experiment is good look at other papers
- Relate it to other experiments a bit
- Tabular problems?
    - Think of relative features
    - Function approximation
- More thought given to addressing concern
- Don’t worry about trying all the algorithms, restrict ourselves
    - Focus on one question
    - Q-Learning

Reference:
Deep learning book – Ian Goodfellow etc.
Standford online CNN course

Previously trained networks (probably don’t need):
Google-net
Alex-net