EVOLVING METAPARAMETERS OF DEEP Q-LEARNING AGENTS IN A PREDATOR/PREY AGENT-BASED MODEL

Leif Rasmussen & Can Gurkan

Department of Computer Science
Northwestern University

ABSTRACT

Multi-agent reinforcement learning systems have garnered attention recently and have been used to solve difficult problems in various settings, such as Atari games and Go [1, 2, 3]. In this study, we have explored an evolutionary multi-agent system with Q-learning agents. We developed a predator/prey agent-based model where each agent has a deep Q-learning network. The network meta-parameters are evolved from generation to generation based on how well each agent learns to perform in its lifetime. We found that the predator and prey agents evolved different network structures and learning parameters in this specific setting. It appears that prey agents benefit from larger networks compared to the predator agents.

1. INTRODUCTION AND PROBLEM DESCRIPTION

Evolution is the only process that has been able to successfully produce general intelligence. An essential feature of evolutionary processes is that they act on agents that are embedded in dynamic environments. In such environments the fitness of agents is determined by their ability to interact with both the static elements of the environment as well as other agents upon which evolutionary processes are concurrently acting. In this study, we have embedded artificial neural networks into agents that are themselves embedded into environments in which they interact with other agents in a competitive way and replicate differentially based on some criteria of success. The agents gain lifetime learning using reinforcement learning. Additionally, evolutionary processes that help shape the meta-parameters of their artificial neural networks from generation to generation. We impose a fitness cost on the agents corresponding to the size of their artificial neural networks to gain insights about trade-offs between complexity and performance. Ultimately, we aim to arrive at a set of techniques for designing evolutionary learning agents that can be used to explore the evolution of intelligence as well as the space of potential behavioral equilibria in different game theoretic scenarios that are not mathematically or computationally tractable.

leif@rasmussen.com & gurkan@u.northwestern.edu

2. RELATED STUDIES

Previous work in multi-agent reinforcement learning begins with using self-play strategies in mastering games like Go, chess and shogi [3] as well as in learning to play two player video games [1, 2]. Other more relevant work involves investigating reinforcement learning agents in a multi-agent setting [4]. Previous work regarding the evolutionary aspect of this study includes growing Artificial Neural Networks using genetic algorithms [5], and training neural networks using genetic algorithms [6].

3. INTELLECTUAL CONTRIBUTION

Both multi-agent reinforcement learning systems and evolutionary neural nets were studied previously as mentioned above, but as far as we know, studies that combine the two areas are relatively few. We believe the techniques we develop during the course of this research have the potential to be used in many domains. The study of complex systems involving adaptive agents spans a number of different academic disciplines. Efficient techniques for developing learning agents will be useful in understanding the nature of intelligence as well as creating computational models of economic processes, socio-cultural processes, ecological processes, etc. Furthermore, these techniques may also be used to create agents capable of practical tasks by having them play/compete in simulations with other learning agents.

4. DESCRIPTION OF THE MULTI-AGENT SYSTEM AND LEARNING

4.1. Multi-agent Model

For this work, we built a simple predator/prey model in which members of the predator species (hawks) survive differentially based on their ability to capture members of a prey species (mice) which in turn will survive based on their ability to evade predators. Both the predator and prey agents use deep Q-learning to approximate state-action-values and determine their actions. The agents have partial observability of their environments. Thus the input state is a real-valued

vector of length 26 which captures the scaled distance of the prey and predator agents within twelve cones with 30-degrees of vision fanning out 360 degrees around the agent. Each cone identifies if there are any predator or prey agents in the 30-degree cone segment of radius r. The actions available to the agents in each state are turning by 20 degrees to the left or to the right or moving forward in the model by 0.1 of a unit.

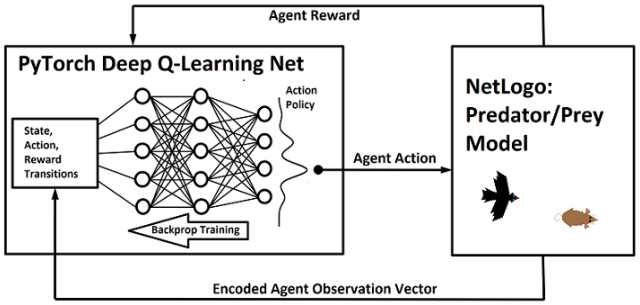


Fig. 1. The multi-agent reinforcement learning framework used in our model.

4.2. Q-learning

Using Q-learning we try to find a state-action value function for each agent which gives us a value for performing each action in the set of possible actions available to the agent given a state. The agent's policy is then determined by choosing an action in a given state based on the values of the state-action function. We train a deep Q-network (DQN) to approximate the state-action value function by passing reward signals associated with the actions in the agent's environment into the deep learning model. Catching a prey for a predator constitutes a +1 reward, while getting caught by a predator for the prey constitutes a -1 reward. We use stochastic batch gradient descent to nudge our deep neural net toward approximating a good state-action function for the agent to use to make decisions in the model. We keep track of the state-action-reward transitions made by the agent in each time step. We then use the following as our target Q (state-action function),

$$Q_{target}(s_{last}, a_{last}) = R + \gamma \max_{a \in A} Q_{actual}(s_{current}, a).$$

4.3. Evolution of metaparameters

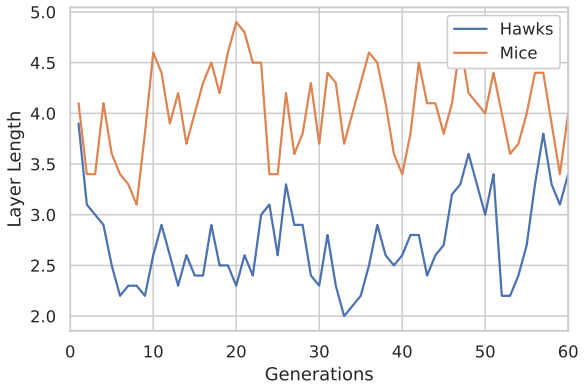
A set of predator and prey agents each initialized with Deep Q-Networks (DQNs) with randomized meta-parameters and randomized weights are generated at the start. The meta parameters are as follows: number of layers in the DQN (bound between 2 and 7,) and the number of nodes in each layer (between 4 and 136, discounting rate, learning rate, batch size, and epsilon (exploration v.s. exploitation) to be optimized via evolution. These agents play a game with each other for a certain amount of time. Predators are rewarded positively for capturing prey and prey are negatively rewarded for being captured respectively. Prey are spawned to random coordinates after being captured to avoid the predators from continuously recapturing the same prey. During this game, agents adapt by updating the weights in their DQNs using batch memory learning. At the end of these play rounds, the predator and prey agents are ranked based on their point minus some cost for the number of parameters in their DQNs. A new population is then initialized using the meta-parameters of a predecessor with a probability proportional to the rank of that predecessor. Random mutations are introduced to the meta-parameters during this process, which is a model of generational learning. Mutations are normally distributed, so larger mutations (such as adding large layers) will be rare compared to small mutations (such as adding an additional unit to a layer). This process is repeated for a set number of generations and the performance changes of the agents in each successive round are recorded. For the sake of computational tractability we restrict the space of possible architectures of DQNs to prevent the number of parameters from becoming too large-although we hope that Occam's razor serves to push the size of the DQNs down to the minimum functional configuration.

5. SOFTWARE

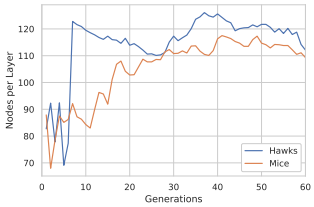
NetLogo is a programming language and development/modeling environment for designing agent-based models [7]. We built a NetLogo simulation of the predator/prey game. We created a Pytorch agent class which implements q-learning and batch memory and embedded it into NetLogo agents using the NetLogo Python extension. In making this model we have created a way of encoding the location of entities in the model into one dimensional vectors.

6. RESULTS

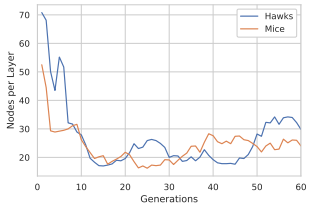
We ran our model for 60 generations where each agent in each generation trained for 30000 time steps. Due to the significant computational cost of our model we weren't able to perform any repetitions, so the statistical significance of our results are low. We used the same agent-based model with agents that act completely randomly as a baseline to compare the behavior of our learning agents with at the end of our experiments as a sanity check. We observed some minor qualitative differences in the behavior of the agents after the end of training rounds. We haven't observed any cooperative dynamics emerge, such as flocking (for the prey) so that they can avoid predators better, or flanking (for the predators) so that they can corner prey. We observed how the size of the networks, and learning parameters change over the generations as shown below.



(a) Mean Layer length

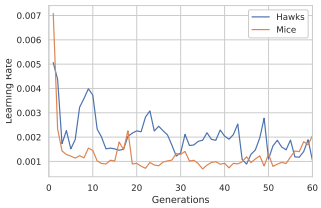


(b) Nodes in Primary Layer

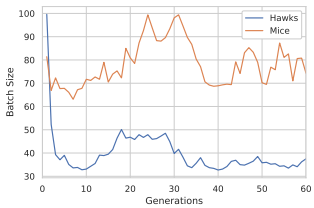


(c) Nodes in Secondary Layers

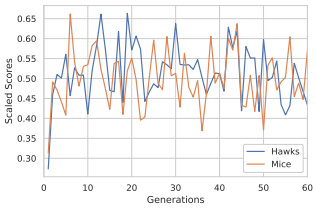
Fig. 2. Evolution of the Network Structure Metaparameters over the Generations. The primary layer is the first layer of the neural net while the secondary layers are the subsequent layers after the first layer.



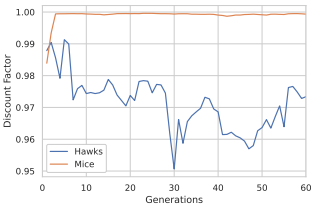
(a) Mean Learning Rate



(b) Mean Batch Size



(c) Mean Scaled Scores



(d) Mean Discount Factor

Fig. 3. Evolution of the Learning Metaparameters over the Generations.

7. DISCUSSION

As can be seen in Figures 2 and 3 above, The predator agents (depicted in Blue) seem to evolve smaller neural networks compared to the prey agents (depicted in orange). The number of nodes per layer are fairly similar in both species, but the prey agents have longer networks. This indicates that the predator agents don't receive a large enough advantage from having large networks compared to the prey agents. We can observe that the prey agents also prefer a larger batch size which means that they sample a larger number of state, action, reward memories at each time step. This is likely needed due to the larger brains that they have. The predator agents have evolved lower discount factors which makes them value only more imminent rewards, this could be interpreted as a more instinctive behavior. The prey agents evolved discount factor close to 1, which indicates that they are more future oriented. Both of the types of agents seem to prefer low learning rates but the prey evolved slightly lower learning rates compared to predators which indicates that they prefer to not learn new knowledge, which makes sense since the environment is fairly stable.

8. CONCLUSION AND FUTURE WORK

In conclusion, we observed that the different species in our predator/prey model evolved different network structures and learning parameters suggesting that the prey species had more to gain from having more cognitive functionality compared to the predators. We hope to use some portion of the software we developed and some of the experimental techniques in this project in our future research. We aim to put q-learning agents in more complex environments with larger state and action spaces and with less computationally tractable equilibria. In the future we would also like to use more computing power to conduct a wider range of experiments and get more robust results.

9. REFERENCES

[1] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell, "Agent57: Outperforming the atari human benchmark," 2020.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[3] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017.

[4] Peter Sunehag, Guy Lever, Siqi Liu, Josh Merel, Nicolas Heess, Joel Z. Leibo, Edward Hughes, Tom Eccles, and Thore Graepel, "Reinforcement learning agents acquire flocking and symbiotic behaviour in simulated ecosystems," *Artificial Life Conference Proceedings*, , no. 31, pp. 103–110, 2019.

[5] Kenneth O. Stanley and Risto Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[6] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," 2018.

[7] Uri Wilensky, "NetLogo," http://ccl.northwestern.edu/netlogo/, Northwestern University, Evanston, IL, 1999.