An Application of AI
Pacman Capture the flag

Artificial Intelligence
COSC1125/1127

Louis Stekhoven-Smith
S3530225
Paul Spende
S3706178
Ran Lu
S3583185

School of Science
Royal Melbourne Institute of Technology
Australia
27.05.2019

1 Experiments/variations and lessons learned

1.1 Approximate Q-Learning

The initial goal was to apply approximate Q-Learning to the 'Pacman capture the flag' problem space. During experimentation, it was found that several key issues have made a practical application challenging to achieve.

The initial issue discovered was the Natural Reward function of winning is only received after many actions are taken. This makes it difficult for the Q-Learner to know what state and action lead to the reward and as such results in getting stuck in local minima.

Creating a reward function that is "better behaved" is a hard problem to solve. During attempts to create such a reward function, it was found that certain assumptions that seemed logical at the surface level would result in undesirable behaviour. An example of this was adding eating food to the reward function. Resulting in the AI eating food and then promptly killing itself so that said food could be picked up again.

Furthermore, the feature space and its value is not easily recognisable. Many features created would conflict with others. An example of this was the conflict between scoring and distance to food. The agent would learn to move toward food as it lost games; however, the agent moved away from food when it scores. This conflict could be overcome by representing features in a more complicated way such as combining distance to food with the amount of food currently carried. Nevertheless, in a complex problem space, the value of these features and dependances were found to be increasingly difficult to identify.

These issues have highlighted the benefits of using machine learning and inverse reinforcement learning to find the value of the feature space and to help find a suitable reward function. Due to time limitations, the team was not able to implement such techniques.

1.2 Mini-Max

Applied to the problem of evading enemy ghosts. The algorithm would decide its next best move using an evaluation function consisting of both ghost distance and food distance. However, even when using depth limited and alpha-beta pruning optimisation techniques, it was found the algorithm consumed excessive computation time. For this reason, the technique was not used, if more time was available using mini-max in a heavily restricted problem space is believed to assist in reducing computation time to a usable level.

2 Applied AI Techniques

2.1 Approximate Q-learning

To overcome the issues identified with Q-learning the problem was simplified to what an aggressive agent's goals are. Such as getting food, scoring points and avoid ghosts. Using this approach in combination with reflex features meant the agent was able to improve its performance and was relatively more straightforward to develop. Though, the behaviour achieved was limited as it did not take into account the overall game state and as such could not achieve long-term strategies.

2.2 AStar Search

A subset of problems was identified in which AStar could be applied. Applying Astar to the problem of finding the best path back to the border when a ghost is within three moves from the agent, proved fruitful. This solution was achieved by using heuristics to calculate the distance to the border, food picked up and distance to ghosts. As the ghost locations changed after each move, only the first action from the resulting path was used, and thus recalculating the path after each turn was needed.

2.3 Sonar Ping

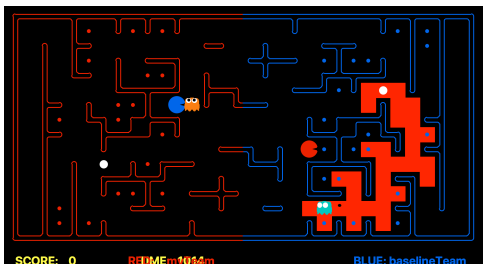

Figure 1: Sonar Ping

It was identified that all maps and enemy spawns are mirrored to ensure fairness in play. This information allows tracking the possible moves an agent can take from their initial location. Cross-referencing these positions with the noisy distance data, returning only the intersection made possible to track the enemy's general location to a varying degree of precision (Figure 1).

3 Non-AI Techniques

3.1 Dead End Identification

Dead ends have the characteristic of being surrounded by precisely three walls, by plotting the dead ends first, it was possible to check the adjacent positions of each dead-end recursively. If the adjacent position has two walls, then it is on the path to a dead end. If the position has less than two walls, then it must be the entrance to the dead-end (Figure 3). The enemy agents positions and dead-end data are used to determine whether it is safe to enter by checking if the closest ghost is at least twice the distance of the length of the current dead-end.

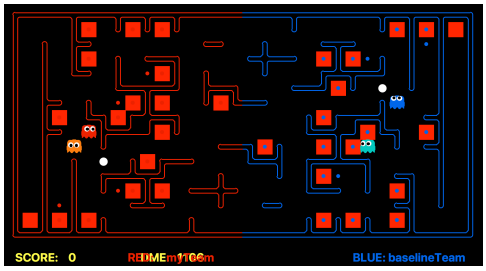

Figure 2: Dead end positions

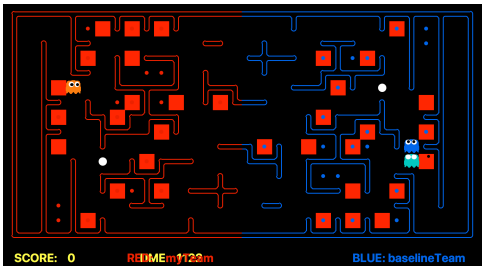

Figure 3: Entries to dead ends

4 Techniques Researched

4.1 Neuroevolution

The team explored the NEAT[1] Framework for application to the Pacman problem. Though during our experiments, we discovered that the search space of fitting an evolutionary neural network was too big for the framework. It did not guide the evolution in the right direction to converge to a sufficient solution.

4.2 Deep Reinforcement Learning

Additionally, we explored the paper "Playing Atari with Deep Reinforcement Learning"[2], looking at how we would feed the image of the "playground" into a neural network that can learn over multiple episodes using the score as a feedback loop. Due to a significant amount of time required to implement and conduct training, it was not feasible to further this approach during this project.

References

[1] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[2] V. Mnih, D. Silver, and M. Riedmiller, "Dqn," *Nips*, pp. 1–9, 2013.