CS188 Final Project Agent Behavior

Team: Niels Joubert, Rohit Nambiar, Micheal Ngo, Tim Chen

Abstract

The final project for CS188 consisted of a capture-the-flag Pacman match of two agents per team matched up with the goal to eat the food on the opponent's side while preventing them from doing so to us. This document is an overview of the approach we took to place third in the final competition.

The course staff provided an defensive and offensive agent, which we used as a kickoff point for the design of our own agents. I observed that the penalty for dying is nothing more than having to start again from the spawn point on our side of the board, and wanted to capitalize on the fact that increasing the score is more important than evading death, thus most of my effort went into writing a strong offensive agent. Concern over the limited amount of computation time provided made an expectimax tree seem like overkill, since we would hardly be able to get more than a 2 level tree out of it. Rather than spend time on expectimax, I[1] concentrated my effort on a reflex agent with a good action evaluation function, while the rest of my team focused on particle filtering and more heuristics for our defensive agent.

1 Offensive Agent Implementation

The duality of agents produced by the ghost/pacman manifestation of agents as they move through the board led me to implement a *nonlinear evaluation function* that varied the weights of features depending on the state of the agent. The features for the agent consisted of:

1. **The score**, with a positive weight of 100, to encourage eating food.
2. **The distance to food**, with a weight of -1, to encourage moving towrds food.
3. **The total food**, with a positive weight of -10, to encourage eating the last food pellet.
4. If the agent is a pacman, on the attack in enemy territory:
   (a) **The distance to the nearest visible enemy ghost**
   If the ghost is within 10 moves, this feature is weighted to 600, else this feature is weighted to 1. Our agent will thus attempt to evade agents if the are in close proximity. This prevents it from reacting to agents when there is a wall between the enemy and itself.
   (b) If we are in the process of evading an agent, we prefer an action that turns us back to a ghost.
5. If the agent is a ghost, still in our own territory on the way to eat food:
   (a) **The number of opponents**, with a weight of -100, thus giving strong preference to a state that eats an invader.
   (b) **The distance to an invader**, with the low weight of -5. Thus we will attack incoming invaders as we move towards eating food.

We spent considerable time experimenting with weights to get adequate behavior, especially the balance between attacking enemies in our territory to aid our defensive agent, and minimizing the amount of food in enemy territory. The weights is also tuned to encourage self-sacrifice if it increases the game score.

[1] I, in this case, being Niels Joubert

2 Defensive Agent Implementation

We considered improved strategies for our defensive agent, as well as implement particle filtering to track enemy agents.

Particle Filtering allowed us to localize enemy agents according to the sonar readings provided by our agents. We could also take advantage of the game by finding the spawn points of enemy agents and the observability of enemy agents when they are within close range to our own. Unfortunately, due to limited time and threading inconsistencies, our particle filtering did not make it into the final agents on time for the competition.

Strategies to increase the defensive agent's performance included exploiting the layout of the board and modifying the weights and features it used to evaluate moves. We noticed a significant increase in performance depending on where the defensive agent spawns, since one of the locations on the board provided easy access to the top of the enemy territory while the other location was more centralized to the food. Contrary to immediate intuition, the spot best suited for defensive spawning is the topmost spot, which guards the alleyway often used by offensive agents. Spawning the offensive agent in a more centralized spot also allowed for more randomness in the path the offensive agent took to reach food, fooling enemy defensive agents.

3 Conclusion

The brashness of our agents paid off and allowed us a third place, with several ties and wins against the first and second place teams. If a rematch were to occur, we would prefer to have particle filtering working, since our defensive agent suffered from lack of knowledge about the position of invaders. Lastly, Q-learning to improve the weights on our features would probably increase the ability of our offensive agent, since it exhibited brashness beyond what was good for the game, often sacrificing itself in the hopes of increasing the score when a more defensive move could have led to survival. Lastly, we would have liked to learn and respond to the behavior of the specific team we were playing against, and we considered Q-learning as a way to adapt to our opponents. Again, this remains to be explored beyond the course.

Thank you for an interesting and exciting class!