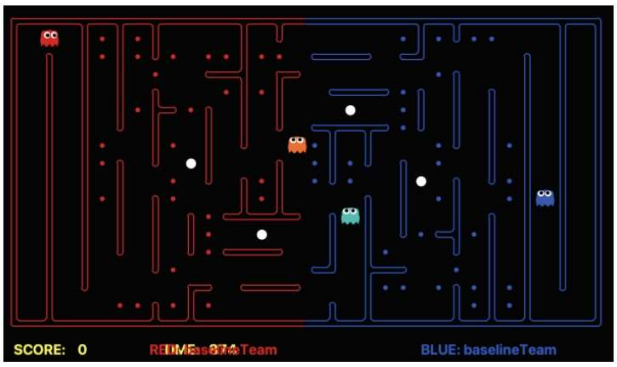
COSE361(03)-Final Project 2/2
Pacman algorithm minicontest 2

Wonseok OH

Abstract

It has various ideas for Pacman game algorithms. We put the best implemented algorithm as the best case, and then check the score by comparing it with other baselines. When implementing Pacman algorithms, various search methods and learning methods are used, and each algorithm is assigned a different color of Pacman when checking the results. The one with the higher score wins. The goal was to get a higher score than the opponent by creating a Pacman that works in various ways.



1. Introduction

- This mini-contest involves a multi-player capture-the-flag variant of Pacman, where agents control both Pacman and ghosts in coordinated team-based strategies. Your team will try to eat the food on the far side of the map, while defending the food on your home side.

- I think it is mainly about comparing Pacman's movements with mine in baseline. Thus, Pacman's movements were identified in the baseline, and Pacman's movements were devised by adding ideas to them.

- The rest is the same as the last assignment, which has a score on the pallet, is terminated when the destination state is reached, and the last value of the reward is paid back. If this is the case, the game will be over, and the results will be based on who won by comparing the scores given.

2. Methods

First of all, baseline picked up three things and added his own format. It consists of your baseline 1, your baseline 2, and your baseline 3. Your baseline 1 added the value of the existing baseline. From here, the offensivereflexagent is entered and the defensive reflex agent is added. In addition, the choiceAction process included a correction value for maxValue, and the getSuccessor section simply stated the position. The evaluation section includes features and weights.

OffensiveReflexAgents is A reflex agent that seeks food. This is an agent we give you to get an idea of what an offensive agent might look like, but it is by no means the best or only way to build an offensive agent. Features, successors, and food lists are further designated. DefensiveReflexAgent is A reflex agent that keeps its side Pacman-free.
Again, this is to give you an idea of what a defensive agent could be like. It is not the best or only way to make It's a suchan agent. Likewise, features and procedures were put in, and my state and my postion were included to complete the coat.

Your baseline 2 adds features from your baseline 1. I thought this result would compensate Pacman for the loss to some extent. Looking for Feature in advance, I assumed that the result would be better, resulting in a better winning rate.
Your baseline 3 is composed of best cases. I tried to improve the results by learning Pacman through Q-Learning.

3. Result

The results were submitted after creating 10 csv files and re-veraging them.

Naturally, your baseline 1 showed the lowest winning percentage and score because it was like baseline. Your baseline 2 speculated that the results would be dominant with information on features, but the results were non. This determined that the result of the code was not perfect. In other words, giving information about features naturally meant that the rest of the information should be given together.

Naturally, your baseline 3 results came out best and I was able to designate this result as the best case. This is due to the principle of reinforcement learning that results are improved when q-learning is conducted through feature verification.

4. Conclusion

As I said earlier, the results of learning through reinforcement learning were also shown in terms of Pacman's score and winning rate. I understand this part explicitly but I think Pacman's result could prove it more clearly.

5. Free Discussion

What is reinforcement learning and why did we use it for the Pacman algorithm?
Reinforcement learning is learning which action is optimal in the current state. Rewards are given in the external environment each time an action is taken, and the learning proceeds in the direction of maximizing these rewards. And these rewards may not be given immediately after action (delayed rewards). For this reason, the difficulty of the problem increases significantly compared to supervised/unsupervised learning, and the challenge of confidence allocation related to properly rewarding the system still afflicts researchers. One can think of making artificial intelligence for games. The current deployment of the enemy's horse from chess becomes the State, and the action is which horse to move. If you catch your opponent's horse, you will be rewarded, and it may take time for the opponent's horse to move far away, so there may be occasions when the opponent's horse is not immediately rewarded. It's even tactically profitable to catch the horse, but it's disadvantageous, so you can lose the game when it's over. (Delayed Compensation). Therefore, reinforcement learning requires action to be selected to maximize the sum of rewards, including those to be obtained later, even if the immediate rewards are a little small, and players who act do not know what actions maximize the sum of those rewards.