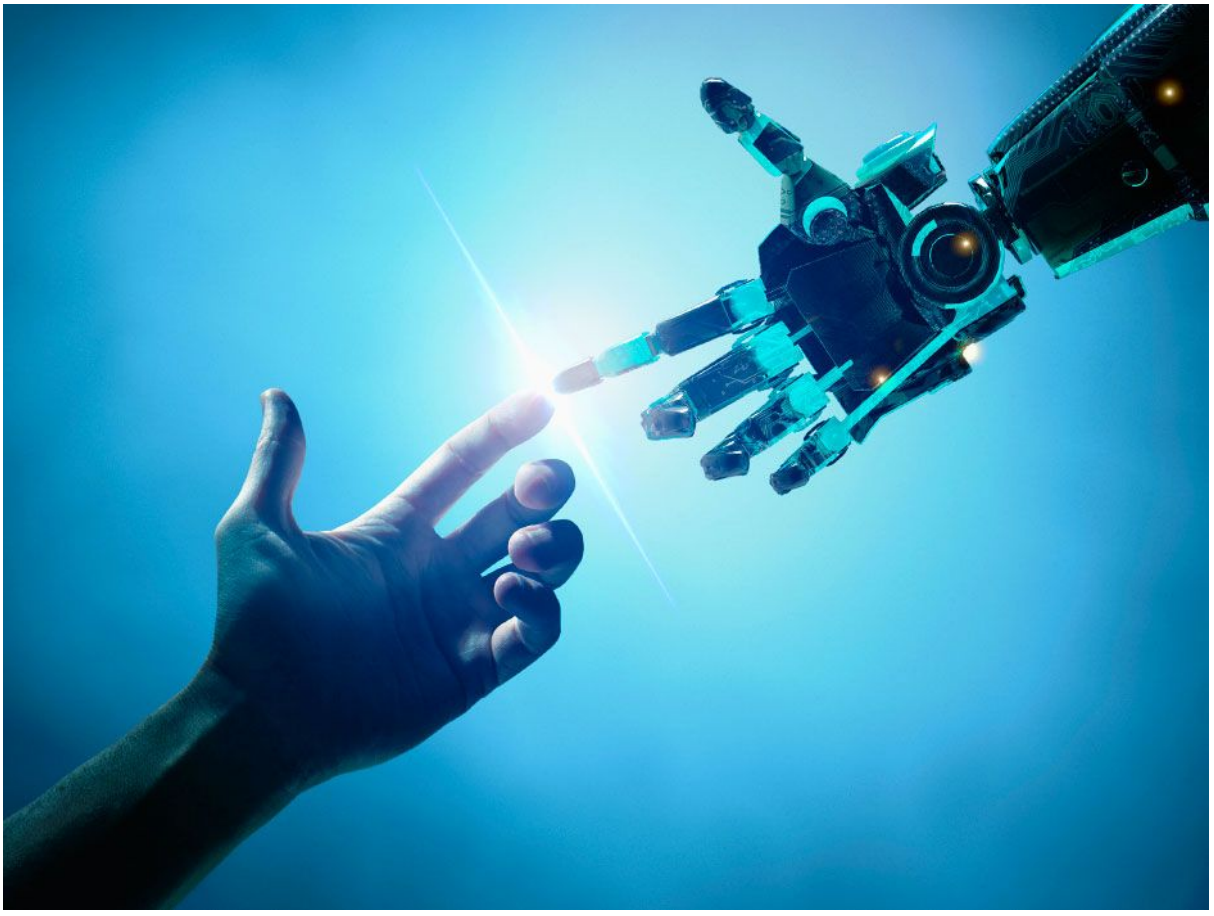
COSC1125/1127 Artificial Intelligence

Project 4, 2018

Contest: Pacman Capture the Flag
Report

Team name -- hopedreamer

Team member:
s3569974 Ning Nan
s3512592 Chao Geng
s3558075 Yang Ding



Introduction

The purpose of this project is to implement a Pacman Autonomous Agent that can play and compete in the Pacman Capture the Flag tournament. In order to get a good result of the tournament, we will give our agent the ability of offence as well as defense which are based on AI techniques, and our agents will have the ability to determine whether it is a good time to attack or defense based on different situations. For our offensive agents, we will give them the ability to find and eat food from opponent's side without been caught by opponent's pacman and bring the food back to get score. For our defensive agents, we will give them the ability to defense any incoming opponents, and protect our foods. There are a lot of methods available for us to implement pacman agents that have the ability we desired. After our discussion and practice in the project, we decide to build a mixed agent based on Approximate Q learning method and minimax method.

Minimax agent(defensive agent)

Design idea

Minimax method is applied to our defensive agent. The idea is from the multiagent project. We can use minimax algorithm to backtrack enemies' action, then choose best and minimum loss method for agent. Our main strategy was that defensive agent turn around 3 foods, which closest to the middle position. If agent found invader, chase and eat it.

Approaches taken

Defensive agent return action base on three factors, main factor is that check if detect enemies. Another factor is based on our foods, this factor have supporting function. That last factor make agent go to food in the middle, when there are not invarder.

Challenges

1. At the start of the design, we did not notice getWall method. We wanted to make defensive agent go to middle position. Only set middle height and middle width. It had fails on some maps, when we tested for different maps. We realized need to make agent avoid the walls.

2. We initial designed the agent , we setted the search depth for reaction is 2. However, if there are two enemies' agents nearby, the reaction time for defensive agent is very slow. In the contest, there were 57 failed contests. We checked the logs, the all fails due to defensive agent take too much time to choose action.

17	hopedreamer	343	105	28	89	222	57	905

After the contest, we realized the problem and setted the depth to 1. The failed contest reduced dramatically to 0. The problem was fixed.

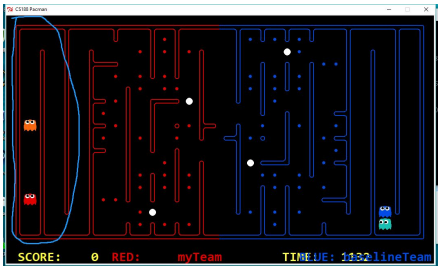
9	hopedreamer	399	128	15	57	200	0	699

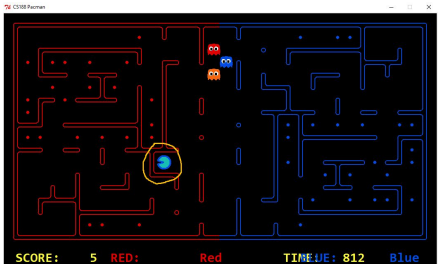
Advantage

Defensive agent can patrol a nearby area, where three foods are closest to the middle position. After several contests, we find out these three food have high probability to become enemies' first choice. Therefore, defensive agent can protect our food quite efficiency. It also could eat invader in time, when defensive agent detected invader.

Possible improvement

1.The agent went to middle position take long time, enemies' agent had enough time to eat our foods for some maps,which have long 's' paths.





2. Sometimes defensive agent stay with enemies' defensive agent at middle position, when enemies' offensive agent was eating our foods.

Approximate q learning agent(offensive agent)

Approximate Q learning is applied to our offensive agent as it is a reinforcement learning technique used in machine learning. To win in the contest, simple if-else is not enough, so we choose to use reinforcement learning techniques. First, we tried to use just Q learning with Q-Tables, but we soon realized that is impossible for us to hold so large table in the memory and changed to Approximate Q learning. It satisfies our requirements which does not need too large space, does learn and after trained, makes right decision on similar states.

Most of our Approximate q learning codes follows the pseudo code from the lecture and previous projects. Things we customized are the features and rewards.

Features: **Food -** Higher feature if closer to the closest food.
Capsule - Higher feature if closer to the closest capsule.
Enemy - Higher feature if closer to the closest ghost that this agent can see.
Return Home - Higher feature if closer to home and carrying more foods. (numFood * 1/distanceToHome)

Rewards: **Reward -** Eat Food, closer to food, eat capsule, score added, closer to home if carrying many food.

Punish - Going away from food, Eaten by ghost.

Above is how we apply Approximate q learning in the offensive agent. Features are used to represent the states, and rewards are used to how to judge the right or wrong action then we update the weights every action. Based on these, after doing a lot of trainings, the agent finally can learn how to choose action based on states.

Trainings

1. First we do 15 trainings with the stationary agent with a high exploration rate. Thus we can get better and better weights on food, capsule and return to home. That is for giving the agent a start weight so it can benefit the coming trainings with enemies.
2. After step 1, our agent can go for food and return home when carrying food. And then, we choose baseline team as enemy, as it can help us train our enemy weights and not too hard for our agent. We did 20 trainings on this. And then our agent can make the almost right action. But still have problems.
3. Then we use one enemy agent which has only one defend agent working using alphabeta method. In that way we can maximize training result of our agent as we are training offensive agent and we only care about enemy ghost. We did 50 trainings on this, and fix a lot of things during the trainings, such as the bugs, reward and features balance.
4. After these trainings, our agent performs well. And then we just adjust some feature policy better based on the performance in the context. And stop updating and exploration rate.

Challenges

1. During our trainings, the most common challenges are the agent start to hover between two points. That is happened because unbalanced features and rewards. We did a lot of work on balancing them.

2. Weights can become infinite after some trainings, that causes exception during runtime. Then we found it is the difference becoming too large, and then we adjust the features to a much smaller value, that problem was fixed.

Possible improvement and disadvantages

1. Agent may go to the dead corner, then eaten by ghost. We can fix that by calculating the depth of corner, then avoid to go inside while ghost is close.
2. Agent has not implemented noisy distance, using noisy distance we can choose a better route and avoid to meet ghost too early.
3. Agent can help teammate to eat enemy in our side. That makes better defence.
4. Agent may need to get away from enemy pacman when it is scared.

Performance

In the final result our agent can beat staff medium stably, and can win at least 3 of 5 against staff top. The offensive agent, can value between food and ghost correctly in most situation. It can eat food without ghost chasing and go away from ghost even there is food around (eat that food may cause the pacman be eaten), decide when to go home when carrying food and eat capsule first. And the main reason that pacman was dead because it went to one dead end. And for the defend agent, it can do its job correctly when pacman around, however, enemy sometimes can rush to our side without being observed by the defend agent.

Conclusion

In conclusion, based on the competition results from the Pacman Capture the Flag tournament, Our team are satisfied with the agent we built as our agent meets most of the ability that we desired before the task, although it is not a perfect agent and there are still some part that we can possibly improve but we all gained a lot techniques and knowledges of artificial intelligence as well as machine learning. We started the project quite late, if we can get more time to do it, it must be much better.