Assignment 4 for COSE361-03

Abstract

이 문서는 COSE361-03의 4번째 과제 minicontest2의 제출물이다. 여기에서는 Reflex Agent를 구성하여 주어진 상황에 대응하는 3개의 Agent를 만들었다. 세 Agent들은 여러 모드를 지니며, 모두 약간의 우위를 계속 유지하는 전략을 채택하고 있다. Agent 중 세 번째 Agent(your_baseline3)가 가장 좋은 성능을 보여 your_best Agent로 채택되었다.

1. Introduction

COSE361-03 Pacman minicontest2는 주어진 2개의 Pacman-Ghost Agent들을 조종하여 상대 팀보다 더 높은 점수를 얻도록 하는 적대적 게임이다. 여기에서, minicontest2에 참가할 세 Agent를 다룰 것이다. 이들은 모두 Reflex Agent이며, 각각의 특징을 가지고 있다.

첫 번째 Agent는 공격과 수비로 나누어진 2인조 Agent 팀으로, 공격 Agent가 두 가지 모드를 지녀 공격을 하다가 상황에 따라 수비 Agent처럼 행동한다. 두 번째 Agent 역시 공격과 수비로 나누어진 팀으로, 공격 Agent가 수비 모드로 전환되면 수비 Agent의 행동 역시 변화한다. 마지막 Agent는 두 개의 같은 Agent의 팀으로, 기회주의자, 공격, 수비의 세 모드를 오가며 상태가 변화하는 Agent이다.

2. Methods

세 Agent는 모두 공통적으로 Reflex Agent이다. 그 말은, 이 Agent들은 상대 Agent의 행동이나 환경의 변화를 예측하지 않고, 오직 현재 환경에 대한 반응으로만 행동한다는 것을 의미한다. 이는 계획적인 Agent를 만들지 못한다는 것을 의미하지만, 다음의 장점을 지닌다.

첫째, 계산 시간이 짧다. minicontest2는 계산 시간에 따라 행동을 하며, 1초 안에 다음 행동을 반환할 수 있어야 한다. 따라서, 행동의 빠른 반환은 더 많은 행동 기회로 이어지며, 이는 적대적 게임에서 큰 이점이 된다.

둘째, 상대의 행동을 예측할 필요가 없다. Reflex Agent들은 상대의 행동에 영향을 받지 않는다. 이는 Agent가 비효율적인 행동을 하는 원인이 될 수도 있지만, 동시에 상대 Agent의 행동 패턴을 파악하기 어려운 minicontest2에서 이점이 될 수도 있다. 부정확한 예측에 기반하여 비효율적으로 작동하는 것을 방지할 수 있기 때문이다.

또한, 공통적으로 세 Agent들은 모두 여러 모드를 가질 수 있다. 이는 getFeature() 함수 외에 getDifFeature() 함수로 구현되었다. 즉, Agent들은 상황에 따라 다른 Feature와 Weight를 사용하여 행동하도록 만들어져 있다.

그렇다면 어째서 이런 전술을 선택하였는가? minicontest2의 점수 산정 방식은, 게임 점수가 단 1점이라도 앞서면 3점을 얻고, 그렇지 않으면 점수를 잃거나 덜 얻는 방식이다. 즉, 승패 자체만이 중요하고 승패의 격차는 중요하지 않다. 따라서 점수가 앞서는 동안은 수비적인 Agent처럼 행동하고, 그렇지 않은 동안은 공격적인 Agent처럼 행동하면 효율적으로 움직일 수 있다.

2.1. your_baseline1

첫 번째 Agent들은 하나는 공격, 하나는 수비를 담당하는 Agent이다. 전술했듯이, 공격 Agent는 두 가지 모드를 지니는데, 첫 번째 모드 중에는 일반적인 공격자로서의 Feature와 Weight를 사용하지만, 두 번째 모드 중에는 수비 Agent처럼 행동하면서 상대 Agent의 침입을 막는 Feature와 Weight를 사용한다.

그렇다면 공격 Agent가 언제 수비 Agent로 전환되는가? 첫째로, 점수가 일정 이상 앞설 때이다. 전술했듯 점수가 앞서면 승리를 굳히고 있으면 되므로, 수비 모드로 전환하여 적이 점수를 얻는 것을 막는다. 둘째로, 상대 유령 Agent에 쫓겨 아군 영역으로 도망쳤을 때이다. 이 경우, 상대 Agent와의 거리가 가까울 확률이 높으므로, 잠시 수비 Agent처럼 행동하여 거리를 벌리고, 다른 방향으로 침입하는 전략을 사용한다.

또한, 공격 Agent는 모든 음식을 한번에 운반하지 않고, 유령인 동안에 일정 확률로 한번에 운반할 수 있는 음식의 수를 증가시킨다(최대 4). 이 음식의 수는 Agent가 죽으면 절반으로 떨어진다(최소 1). 이를 통해 Agent가 음식을 모으다가 죽는 것을 반복하지 않고, 약간의 우위라도 더 가져갈 수 있도록 할 수 있다.

2.1.1. REFLEXOFFAGENT

공격 Agent의 평상시(공격 모드)의 Feature는 다음과 같다. 이중 6, 7은 공격 모드인 동안에도 현재 Agent가 유령이라면 침입한 적 Agent에 대항하도록 하는 역할을 하며, 8은 특정 지점에 정지해서 끼이는(Stuck) 것을 방지한다.

1. 음식의 개수
2. 음식이 있다면, 음식까지의 최소 거리
3. 음식이 있다면, 음식까지의 평균 거리
4. 팩맨인 동안, 적 유령이 있다면, 유령과 매우 근접하였는지의 여부
5. 팩맨인 동안, 적 유령이 있다면, 유령과의 최소 거리
6. 유령인 동안, 적 팩맨이 있다면, 적 팩맨과의 최소 거리[1]
7. 적 팩맨의 수
8. 행동이 Directions.STOP이 아닌지의 여부

공격 Agent의 수비 모드 Feature는 다음과 같다. 이중 2는 수비 모드인 동안에 상대 영역에 있다면 최대한 빨리 도망치도록 하는 역할을 하며, 5는 공격 Agent가 수비 모드 중에는 캡슐들을 지키는 것을 목표로 삼도록 한다.

1. 자신이 유령인지의 여부
2. 팩맨인 동안, 적 유령과의 최소 거리
3. 유령인 동안, 상대 팩맨의 수
4. 유령인 동안, 적 팩맨이 있다면, 적 팩맨과의 최소 거리
5. 아군 캡슐이 있다면, 캡슐과의 평균 거리

2.1.2. REFLEXDEFAGENT

수비 Agent의 Feature는 다음과 같다.

1. 자신이 유령인지의 여부
2. 적 팩맨의 수
3. 적 팩맨이 있다면, 적 팩맨과의 최소 거리
4. 적 Agent와의 최소 거리
5. 자신이 공포 상태라면, 적 팩맨과의 최소 거리

2.2. your_baseline2

두 번째 Agent들 역시 공격과 수비 Agent로 나뉜다. 또한, 공격 Agent는 마찬가지로 공격 모드와 수비 모드를 가지며, 수비 모드로의 전환 조건이 조금 달라졌는데, 점수가 1점이라도 앞선다면, 그리고 오직 그 경우에만 수비 모드로 전환한다. 만약 우위가 사라지면 다시 공격 모드로 전환한다. 또한, 공격 Agent가 수비 모드에 들어가면, 수비 Agent 역시 행동이 변화한다. 자세한 변화는 후술한다.

추가로, 두 번째 공격 Agent는 초기화 시 initFood 변수를 지니는데, 이 변수는 출발 지점 기준으로 상대 Agent와의 거리와 자신과의 거리의 차가 가장 큰 음식의 위치를 저장한다. 즉, 최적의 경로로 이동하였을 때, 획득할 확률이 높은 음식의 위치를 저장한다. 이 음식이 위치를 벗어나기 전까지, 공격 Agent는 해당 음식을 최우선적으로 공략한다.

또한, 각 Agent는 담당하는 적 Agent가 있는데, 일부 Feature는 해당 Agent만을 대상으로 계산된다. 이 방식의 장점은 두 적 Agent가 모두 공격 Agent로 활동하거나, 상대 수비 Agent가 아군 수비 Agent를 교란하는 것을 막을 수 있다는 점이다.

2.2.1. REFLEXOFFAGENT

공격 모드에서, 공격 Agent의 Feature는 다음과 같다. 특기할만한 점은, 음식을 하나라도 들고 있다면 아군 영역으로 복귀하는 것을 우선시한다는 점이다. 이는 점수가 1점이라도 앞서면 수비 모드로 전환된다는 특성과 맞물려, 1점 우위를 점한 후 수비하는 전략으로 귀결된다.

1. 음식의 개수
2. 음식이 있다면, 음식까지의 최소 거리
3. 음식이 있다면, 음식까지의 평균 거리
4. 팩맨인 동안, 적 유령이 있다면, 유령과 매우 근접하였는지의 여부
5. 팩맨인 동안, 적 유령이 있다면, 유령과의 최소 거리[1]
6. 음식을 들고 있는 경우, 아군 영역과의 최소 거리
7. 아군 영역으로 돌아왔는지의 여부

수비 모드에서, 공격 Agent의 Feature는 다음과 같다. 4번은 기존 적 Agent와의 최소 거리를 개선한 Feature인데, 수비 모드일 때는 오직 아군 영역에만 머무르기 때문에, 적의 현재 위치가 아니라 적이 침입할 위치까지의 거리를 구하도록 한 것이다.

1. 자신이 유령인지의 여부

2. 적 팩맨의 수
3. 적 팩맨이 있다면, 적 팩맨과의 최소 거리
4. 담당하고 있는 적에 가장 가까운 아군 영역까지의 최소 거리

2.2.2. REFLEXDEFAGENT

일반적인 상황에서, 수비 Agent의 Feature는 다음과 같다.

1. 자신이 유령인지의 여부
2. 적 팩맨의 수
3. 적 팩맨이 있다면, 적 팩맨과의 최소 거리
4. 적에 가장 가까운 아군 영역까지의 최소 거리

공격 Agent가 수비 모드가 되는 경우, 수비 Agent의 Feature는 다음과 같다. 기존 수비 Agent는 두 상대 Agent 중 가까운 Agent를 막으려 시도하는 반면, 수비 모드가 되면 담당 상대 Agent로 이동하여 해당 Agent만을 전담하게 된다. 이는 수비 모드에서는 공격 Agent가 나머지 상대 Agent를 방어하고 있을 것이기 때문이다.

1. 자신이 유령인지의 여부
2. 적 팩맨의 수
3. 적 팩맨이 있다면, 적 팩맨과의 최소 거리
4. 담당하고 있는 적에 가장 가까운 아군 영역까지의 최소 거리

2.3. your_baseline3

세 번째 Agent는, 공격 Agent와 수비 Agent가 명시적으로 정해져있지 않다. 일명 기회주의자 Agent(OpportunistAgent) 라고 불리는 이 Agent들은, 모두 수비를 하다가 상대 Agent의 빈틈을 발견하면 한 Agent만 공격 모드로 전환하여 소량의 음식을 획득한다. 점수가 상대보다 높아진 경우 수비를 고수하고, 점수가 따라잡히면 다시 기회주의자처럼 행동하는 것을 반복한다.

또한, 이 Agent들은 수비 Agent 특성 상 특정 위치에 끼이는(Stuck) 일이 발생할 확률이 높기 때문에, 우위를 확보한 상태가 아니라면 점차적으로 이동하지 않았던 지점으로 탐험하는 특성을 지니고 있다.

2.3.1. OPPORTUNISTAGENT

기회주의자 Agent의 평상시 Feature는 다음과 같다. 기본적으로는 수비 행동을 하되, 최전선에서 수비하는 동안 위아래로 움직이면서 적 Agent의 틈을 노린다. 공격 모드로 전환되는 구체적인 조건은, 이기고 있지 않으며, 두 아군 Agent가 모두 최전선에서 대치 중이고, 상대 Agent와 음식이 존재하고, 상대 Agent가 음식 또는 캡슐에 비해 충분히 멀리 있다고 판단될 때이다. 또한, 지도 상의 모든 부분에 대하여, 해당 위치로 이동하거나 머무른 횟수를 기록하여, 새로운 지점으로 이동하도록 하였다.

1. 자신이 유령인지의 여부
2. 적 팩맨의 수
3. 적 팩맨이 있다면, 적 팩맨과의 최소 거리
4. 적이 있다면, 담당하고 있는 적에 가장 가까운 아군 영역까지의 최소 거리
5. 아군의 최전선에 있는 동안, 이동 방향이 북쪽 혹은 남쪽인지의 여부
6. 공격 Agent로 전환하기 위해 전진해야 하는지의 여부
7. 해당 위치로 이동한 횟수

기회주의자 Agent의 수비 시 Feature는 다음과 같다. 평상시와 다른 점이 있다면, 이 모드 중에는 아군의 점수가 상대보다 높기 때문에, 기회를 보아 점수를 더 얻으려고 하지 않고 오직 방어에만 전념한다는 점이다. 또한, 4번을 보면 알 수 있듯이, 적과 가까운 아군 영역이 아니라 적을 직접적으로 쫓는데, 대치 또는 본격적인 수비 상태에서는 이것이 더 이점이 크기 때문이다.

1. 자신이 유령인지의 여부
2. 적 팩맨의 수
3. 적 팩맨이 있다면, 적 팩맨과의 최소 거리
4. 담당하고 있는 적까지의 최소 거리

기회주의자 Agent의 공격 시 Feature는 다음과 같다. 음식을 들고 있는 경우, 1번 Feature의 값이 줄어드는데, 그 이유는 음식을 하나라도 확보했다면 더이상의 음식을 확보할 필요성이 적어지기 때문이다. 또한, 8번 Feature는 음식을 들고 있는 경우, 복귀하지 못하고 막다른 골목에 들어가 죽는 일을 방지하도록 한다. 음식을 들고 있지 않은 경우에는 작동하지 않는데, 음식이 없는 상태에서는 해당 음식을 먹고 죽어서 위치를 옮기는 게 득이 되는 전략일 수 있기 때문이다.

1. 음식의 개수
2. 음식이 있다면, 음식까지의 최소 거리
3. 팩맨인 동안, 적 유령이 있다면, 유령과 매우 근접하였는지의 여부

Table 1. your_baseline3의 대전기록

상대	승률	평균 점수
YOUR_BASE1	0.9	0.9
YOUR_BASE2	0.8	0.7
YOUR_BASE3	-0.0	0.0
BASELINE	1.0	1.0

4. 팩맨인 동안, 적 유령이 있다면, 유령과의 최소 거리[1]
5. 캡슐의 개수
6. 캡슐이 있다면, 캡슐과의 최소 거리[1]
7. 음식을 들고 있는 경우, 아군 영역과의 최소 거리[1]
8. 음식을 들고 있는 경우, 해당 위치에서 움직일 수 있는 경우의 수가 2가지 초과인지의 여부

3. Results

your_baseline3은 자신을 제외한 모든 Agent를 상대로 0.8 이상의 승률을 보였다. 특히, baseline Agent를 대상으로는 1.0의 승률을 보였다. 전체 승률은 0.675였으며, 평균 점수는 0.65점이었다. 특기할만한 점은, 평균 점수가 매우 낮다는 점인데, 이는 your_baseline3 Agent가 단 1점의 우위라도 확보하고 이를 유지하는 것을 목표로 하는 Agent이기 때문이다.

4. Conclusion & Free Discussion

Pacman minicontest2에서, 적은 점수 우위를 얻고 이를 유지하는 Reflex Agent의 구현에 대해서 알아보았다. 구현된 Agent 중 가장 높은 승률을 보여준 Agent는 your_baseline3, 일명 기회주의자 Agent(OpportunistAgent)로, 공격과 수비 역할을 명시적으로 정하지 않고, 상황에 따라 약간의 점수를 탈취한 후 수비로 이를 유지하는 Agent였다.

Reflex Agent를 구현하지 않고, Planning Agent를 구현한다면 어떻게 할 수 있을까? 아마 많은 Agent들은 공격 Agent라면 음식과의 거리, 음식의 개수 등을 Feature로 사용할 것이고, 수비 Agent라면 팩맨과의 거리, 팩맨의 개수 등을 Feature로 사용할 것이다. 이를 기반으로 가상의 적 Agent를 시뮬레이트하여 Minimax나 Expectimax Agent를 구현할 수 있을 것이다. 다만, 이 경우 컴퓨팅 시간의 절약에 초점을 맞춰야 할 것이다.

초기화 시간이 15초나 주어졌는데, 이를 잘 활용할 수 있는 방법은 없었을까? 주어진 Distancer 객체는 적 Agent를 고려하지 않은 채 지점 간 거리를 계산했다. 각 좌표에 적 유령이 존재한다고 가정했을 때, 지점 간 최단 거리를 초기화 시간에 계산하여 메모리에 유지할 수도 있다. 그렇다면 Feature로 적의 위치나 적과의 거리를 사용하여 간접적으로 Agent의 방향을 이동시키지 않고 메모리에 저장된 값을 기반으로 거리를 잴 수 있을 것이다.

[1]log : 로그를 취한 이유는, 특정 Feature가 값이 작을 때만 큰 영향을 미치고, 값이 커져도 영향을 적게 미치고 싶어서이다