**프로젝트 기획서**

인공지능

글로벌미디어학부

20092518 김민종

**1. Q-learning을 활용한 인공지능 게임**

인간은 두뇌를 통해서 게임의 이해와, 조작법, rule을 습득하고 그에 따른 게임의 결과값을 찾기 위해 시도하고 분석한다. 이와 같이 인공지능의 q-learning algorithm을 사용하여 문제를 제시해 주고, 그 경우에 따른 결과값을 학습을 통해 최종적인 목표에 도달함을 구현해 보고자 한다

**2. Unity 3d를 활용한 강 건너기 게임**

누구나 한번씩 풀어보았던 두뇌게임인 강 건너기 게임. 주어진 목표가 있고, STAGE마다 서로간의 상성이 존재하는 (ex 늑대, 양, 곡식) 요소들을 강 위에 떠있는 배를 이용하여 서로 간의 영향을 주지 않고 강 건너로 옮겨가는 게임이다.

**3. 게임 구현 ( STAGE는 3개정도로 구성 할 예정)**

**STAGE1 늑대, 양, 곡식이 존재한다.**

**조건)**

1) 사람이 없으면 늑대가 양을 잡아먹는다.

2) 사람이 없으면 양이 곡식을 먹는다.

3) 한번에 하나씩 옮길 수 있다.

4) 사람은 무조건 이동해야 한다.

**4. 알고리즘 구현**

**1) 초기스택 = {0,0,0,0};**

**2) Stack[index][4]= [실행횟수][양 or 늑대 or 곡식 or 사람];**

**3) 이동 Stack 0 ->1; ( START = 0, GOAL = 1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | 늑대 | 양 | 곡식 | 사람 |
| Index-1 | 0  Stack[0][0] | 0  Stack[0][1] | 0  Stack[0][2] | 0  Stack[0][3] |
| Index-2 | 1  Stack[1][0] | 0  Stack[1][1] | 0  Stack[1][2] | 1  Stack[1][3] |
| Index-3 | 0  Stack[2][0] | 1  Stack[2][1] | 0  Stack[2][2] | 1  Stack[2][3] |
| Index-4 | 0  Stack[3][0] | 0  Stack[3][1] | 1  Stack[3][2] | 1  Stack[3][3] |
| Index-… | …  Stack[…-1][0] | …  Stack[…-1][1] | …  Stack[…-1][2] | …  Stack[…-1][3] |
| Index-i | 1  Stack[i-1][0] | 1  Stack[i-1][i] | 1  Stack[i-1][2] | 1  Stack[i-1][3] |

**제약 조건)**   
1) 사람이 없고 늑대와 양이 있는 경우 : 1100, 0011, 1110, 0001  
2) 사람이 없고 양과 양배추가 있는 경우 : 0110, 1001, 1110, 0001  
3) 중복되는 경우를 제외하면 0001, 0011, 0110, 1001, 1100, 1110의 6가지가 제약 조건이됨

**4) reward 구성**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Reward** | **0000** | **0001** | **0010** | **0011** | **0100** | **0101** | **0110** | **0111** | **1000** | **1001** | **1010** | **1011** | **1100** | **1101** | **1110** | **1111** |
| **0000** | **-1** | **-100** | **-1** | **-100** | **-1** | **-100** | **-1** | **-1** | **-1** | **100** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **0001** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **0010** | **-1** | **-1** | **-1** | **-100** | **-1** | **-1** | **-1** | **100** | **-1** | **-1** | **-1** | **100** | **-1** | **-1** | **-1** | **-1** |
| **0011** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **0100** | **-1** | **-1** | **-1** | **-1** | **-1** | **100** | **-1** | **100** | **-1** | **-1** | **-1** | **-1** | **-1** | **100** | **-1** | **-1** |
| **0101** | **50** | **-1** | **-1** | **-1** | **50** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **0110** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **0111** | **-1** | **-1** | **50** | **-1** | **50** | **-1** | **-100** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **1000** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-100** | **-1** | **100** | **-1** | **100** | **-1** | **-1** |
| **1001** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **1010** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **100** | **-1** | **-1** | **-1** | **100** |
| **1011** | **-1** | **-1** | **50** | **-1** | **-1** | **-1** | **-1** | **-1** | **50** | **-1** | **50** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **1100** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **1101** | **-1** | **-1** | **-1** | **-1** | **50** | **-1** | **-1** | **-1** | **50** | **-1** | **-1** | **-1** | **-100** | **-1** | **-1** | **-1** |
| **1110** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **1111** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |

1) 가능 할 때는 : +100

2) 조건에 해당될 경우 -100

3) 가능 했던 조건으로 다시 돌아갈 경우 +50 (중복을 없애기 위한 조건)

4) 노드가 없을 시엔 -1

**5. 게임 진행**

**자동으로 index(실행횟수) 마다 stack값 저장,**

**제약조건에 해당될 경우 게임 초기화, but Stack값과 index값은 저장함.**

**실행 횟수에 따른 reward를 통한 게임 clear**

**5. 코드**

int stack[100][4]; //초기 스택 크기 설정

int index = 100; //인덱스 크기 설정

int arr[4]; // 배열 크기 설정

void Play(int arr[]) {

for (int i = 0; i<index; i++) {

//START에서

if (arr[3] == 0) {

if (arr[0] = 양 arr[1] = 늑대 arr[2] = 곡식 NO 중 선택) {

arr[4] = 1, arr[1, 2, 3중 선택] = 1;

stack[index][i] = arr[i];

//스택에 배열값 저장. (arr이 1이면 GOAL로 이동, 아니면 START)

If(NO 선택){

arr[4] = 1, arr[1,2,3] = 0;

}

index++;

chk();

}

}

//GOAL에서

if (arr[3] == 1) {

if (arr[0] = 양 arr[1] = 늑대 arr[2] = 곡식 중 선택) {

arr[4] = 0, arr[i, 2, 3중 선택] = 0;

stack[index + 1][4] = arr[4];

stack[index + 1][i] = arr[i];

//스택에 배열값 저장. (arr이 1이면 GOAL로 이동, 아니면 START)

}

index++;

chk();

}

if (현재 Stack의 reward가 - 100 이라면) {

처음부터 재시작;

}

else {

계속진행;

}

}

finished();

}

void chk() {

//제약조건 검사

if (늑대 == 양 && 늑대 != = 사람) || 양 == 곡식 && 양 != = 사람){

reward -= 100;

}

else {

reward += 100;

}

//중복검사

for (int i = 0; i<index - 1; i++{

if (현재 스택에서 다음 이동 스택 == 이전에 있던 스택) {

reward += 50;

}

}

}

void finished() {

for (int i = 0; i < 4; i++) {

if (스택이1111이라면) {

게임종료;

}

else

계속진행;

}

}

int main(void) {

int init[4] = { 0,0,0,0 };//초기스택 설정;

Play(init);

}