게임 환경을 통제할 수 있는 규칙 기반 Semi-Supervised Learning 오목 인공지능 프레임 워크

김선민, 구본우*
경민대학교 게임콘텐츠과
griogrio@kyungmin.ac.kr

Rule based Semi-Supervised Learning Gomoku Game AI Framework for Control Game Environment

Sun-Min Kim, Bon-Woo Gu
Department of Game Contents, Kyungmin University

요 약

게임은 수많은 NPC 와 규칙에 의해 작동되는 가상 공간을 의미한다. 이런 가상 공간에서는 규칙을 엄격히 지키면서 수행되는 AI 를 필수로 요구하게 된다. 하지만 강화 학습 기반의 AI 는 복잡한 게임의 규칙을 온전히 지키지 못하고 예상 밖의 행동을 돌출하면서 이를 해결하기 위한 많은 연구도 수행되고 있다. 본 논문에서는 규칙 기반으로 획득한 오목판의 확률 맵과 학습을 통해 획득한 확률맵 데이터를 병합하여 가장 높은 Value 를 가지는 위치를 다음 수로 반환하는 방법을 사용하였다. 향후 연구에서는 ANN(Approximate Nearest Neighbor)알고리즘을 적극 활용하여, 커널의 State 와 보드의 State 비교를 확률적으로 개선할 예정이다. 본 논문에서 제안된 프레임 워크는 게임 AI 연구에 기여할 수 있길 바란다.

1. 서론

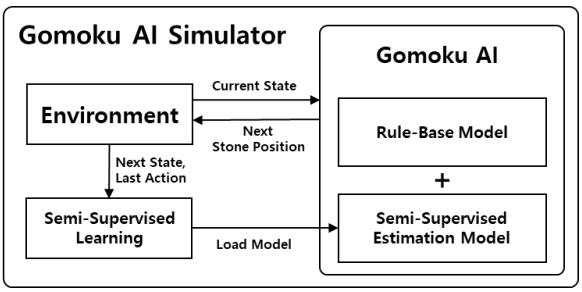
게임은 수많은 NPC 와 규칙에 의해 작동되는 가상 공간을 의미한다. 이런 가상 공간에서는 규칙을 엄격히 지키면서 수행되는 AI 를 필수로 요구하게 된다.[1][2] 최근 Deep Learning 의 우수한 연구 성과로 인해 많은 게임 관련 연구자들도 강화 학습 기반의 AI 를 연구하고 있다.[3][4] 하지만 강화 학습은 복잡한 게임의 규칙을 온전히 지키지 못하고 예상 밖의 행동을 돌출하면서 이를 해결하기 위한 많은 연구도 수행되고 있다.[5]

본 논문에서는 규칙 기반 게임 AI 에 이전 상황을 확률맵으로 학습한 Semi-Supervised Learning 를 적용한 프레임 워크를 제안한다. 본 논문에서 제안한 방식은 게임의 규칙을 최우선으로 고려하며 과거의 학습된 결과로 최적의 의사를 결정한다. 본 논문의 AI 프레임 워크는 오목 게임에 적용하여 실험하였다. 본 논문에서 제안된 프레임 워크는 게임 AI 연구에 기여할 수 있길 바란다.

2. 규칙 기반 준 학습 오목 AI 프레임 워크

규칙 기반 준 학습 오목 AI 프레임 워크는 2 단계를 반복적으로 수행한다. 그림 1 은 본 논문에서 제안한 규칙 기반 준 학습 오목 AI 프레임 워크의 Flow Chart 를 보여준다.

Gomoku AI Simulator 에서는 “Gomoku AI”에서 받아온 수를 저장하고 저장된 오목판의 정보인 “Environment”를 가지고 게임의 시각화를 작업한다.



(그림 1) 오목 AI FrameWork

“Gomoku AI”는 “Rule-Base Model”과 “Semi-Supervised Estimation Model”의 확률맵을 병합하여 최적의 다음 수를 제공하는 오목 인공지능 의미한다. “Gomoku AI”은 “Environment”에서 제공받은 오목판의 정보를 이용하여 인접한 같은 돌의 수를 바탕으로 Value 를 주는 규칙 기반의 확률맵과 이전 게임 기록을 바탕으로 가져온 학습 확률맵을 계산한다. 최종적으로 계산된 각각의 확률맵을 합산하여 가장 높은 Value 를 가지는 위치를 다음 수로 선택하여 “Environment”에 제공한다. “Environment”는 제공된 다음 수의 위치를 통해 오목 게임을 진행하고, 오목판의 정보를 최신화 한다. 승패가 결정된 경우, “Environment”에서 승점 수의 이전의 수를 중점으로 일정한 커널 사이즈를 가지는 데이터

를 “Semi-Supervised Estimation Model”에서 활용할 수 있도록 “Semi-Supervised Learning”을 진행한다.

수도 코드 1:
Semi-Supervised Learning 오목 알고리즘

```
INPUT : BoardArr, WArr, DataArr
OUTPUT : NextPos

ismatch ← True
WArr ← RB_StoneCout()

for i ← 0 to DataArr.size do:
  DataArr ← DataLoad(i)
  ismatch ← True
  for x,y ← 0 to BOARDSIZE do:
    if BoardArr[x][y] is not DataArr then:
      ismatch ← False
    end if
  end for

  if ismatch is True then:
    id ← argmax_{x,y}(DataArr)
   WArr[id] ← WArr[id] + reward
  end if
end for

NextPos ← argmax_{x,y}(WArr)

return NextPos
```

<표 1> Semi-Supervised Learning 오목 알고리즘

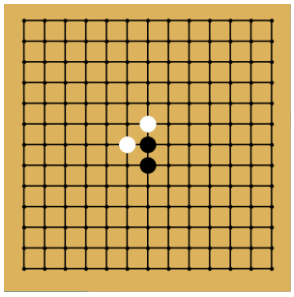
표 1 는 규칙 기반 준 학습 오목 AI 프레임 워크의 수도코드를 보여준다. 본 모듈에서는 기존에 학습한 데이터와 현재의 오목판을 비교하여 과거에 최종적으로 패배하였거나 승리한 순간의 승점을 불러와 현재의 확률 맵에 더해주는 방식으로 현시점의 최적의 수를 찾을 수 있도록 작동하고 있다.

BoardArr 는 “Environment”을 통해 받아온 오목판을 가지는 배열이다. WArr 는 다음 수의 Value 를 가지는 오목판과 같은 크기를 가진 배열이다. RB_StoneCount 는 빈칸이 흑과 백돌일 경우, 각 8 방향의 서로 접한 돌의 개수를 받아 Value 를 담은 반환한다. DATASIZE 는 데이터의 커널 크기를 의미한다.본 논문에서는 커널 크기에 변화를 주고 각각의 실험을 진행하였다. DataArr 는 DATASIZE 만큼의 커널 크기를 가지는 오목 데이터이다. ismatch 는 DataArr 와 BoardArr 가 일치하는지 확인하기 위한 값을 의미한다. reward 는 지난 학습 데이터를 통해 얻는 Value 값을 의미한다. 본 실험에서는 reward 의 값을 10 으로 설정하고 실험을 진행하였다. argmax 는 최종 확률 맵에서 가장 Value 가 높은 위치 값을 반환한다.

3. 실험 결과

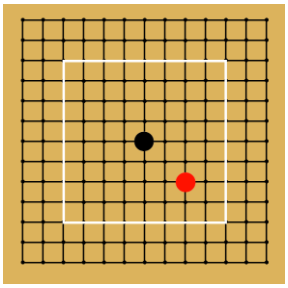
3.1 실험 결과 화면

그림 2 는 C++을 기반으로 OpenGL 라이브러리를 이용하여 제작한 오목 AI 시뮬레이터의 화면을 보여준다. 본 실험에서 OpenGL 라이브러리는 오목판과 돌들의 시각화를 담당하였다. 본 실험에서는 13x13 의 크기의 보드를 기준으로 실험을 진행하였다.



(그림 2) 오목 시뮬레이터

그림 2 에서처럼 실험에서 사용한 게임은 흑 돌을 중앙에 선으로 배치하고 한 턴씩 번갈아 흑과 백 돌을 배치하는 오목 게임이다.



(그림 3) 오목 시뮬레이터

그림 3 는 본 시뮬레이터에서 사용하는 데이터를 설명하기 위한 이미지이다. 본 실험은 그림 3 과 같이 승부수의 이전 수를 기준으로 마지막 수에 Value 를 준 현재 오목판의 돌 정보 데이터를 저장하고 활용한다.

3.2 실험 결과

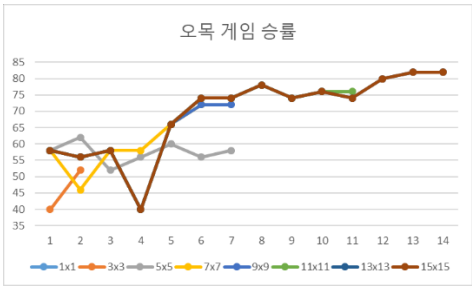
	1x1	3x3	5x5	7x7	9x9	11x11	13x13	15x15
7	58	52	58	72	72	74	74	74
8	58	52	58	72	72	78	78	78
9	58	52	58	72	72	74	74	74
10	58	52	58	72	72	76	76	76
11	58	52	58	72	72	76	74	74
12	58	52	58	72	72	76	80	80
13	58	52	58	72	72	76	82	82
14	58	52	58	72	72	76	82	82

<표 2> 오목 승률 표

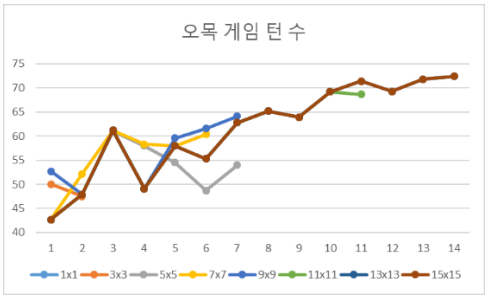
표 2 와 그림 2 는 실제 학습을 진행하여 회차 마다의 게임 승률을 기록한 표와 그 그래프이다. 각 커널의 사이즈 차이에 따른 승률이 기록되었다. 표 2 에서 정체구간은 푸른색으로 표시하였다.

본 실험은 저장하는 데이터의 크기에 차이를 두고 실험을 진행하였다. 실험은 흑 돌을 기준으로 1 회차에 50 게임을 14 회차까지 진행되었고, 각 게임마다 게임의 승 패에 대한 정보를 각 커널 크기에 따라 저장하여 다음 게임부터 받아와 사용하였다.

표 2 의 1X1 의 경우 규칙 기반만 적용한 오목 AI 와 같은 결과를 가진다.



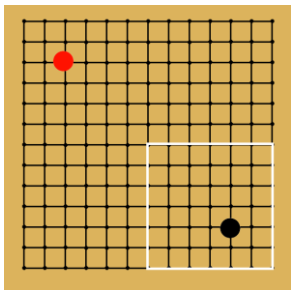
(그림 4) 커널 크기에 따른 승률



(그림 5) 커널 크기에 따른 턴 수

그림 4 과 그림 5 에서 확인할 수 있는 것처럼 평균적으로 회차의 진행에 따라 게임의 승률과 평균 턴 진행 수가 증가하는 결과를 얻을 수 있었다. 우상향하는 게임의 승률과 턴 수를 바탕으로 성공적으로 학습이 이루어 지고 있음을 확인할 수 있었다.

하지만 데이터의 커널이 작을수록 게임의 승률과 턴 수가 변하지 않는 정체 구간이 빠르게 나타난다. 반면, 커널 크기가 큰 게임에서는 정체 구간이 늦게 발생하며, 우상향으로 나타난다.



(그림 6) 커널 크기에 따른 턴 수

그림 6 은 실험의 예시를 시각화하여 표현한 것이다. 검은 점은 데이터의 중점으로 삼는 승부수의 이전 수를 의미한다. 빨간 점은 게임의 승부를 결정짓는 마지막 수인 승부수를 의미한다. 그림 6 와 같이 승부수가 커널의 범위 안에 들어오지 못했을 경우 데이터를 활용하지 못하고 승률이 정체하는 것이다.

하지만 이러한 문제는 그림 6 에서 확인할 수 있는 것처럼 커널의 크기가 증가함에 따라 해결되는 부분임을 확인하였다.

4. 결론

본 논문에서는 게임의 환경을 통제할 수 있는 Rule-Base Supervised Learning 오목 인공지능 프레임워크를 제안하였으며, 규칙 기반과 준 학습을 이용하여 최적의 오목 수를 계산하는 실험을 진행하였다.

본 실험에서는 규칙 기반으로 획득한 오목판의 확률 맵과 학습을 통해 획득한 확률맵 데이터를 병합하여 가장 높은 Value 를 가지는 위치를 수로 반환하는 방법을 사용하였다. 낮은 커널 크기를 사용하는 실험에서는 승률이 정체하는 현상을 발생하였으나 커널의 크기를 증가시킴으로써 문제를 해결하였다.

향후 연구에서는 ANN(Approximate Nearest Neighbor) 알고리즘을 적극 활용하여, 커널의 State 와 보드의 State 비교를 확률적으로 개선할 예정이다. 본 논문에서 제안된 프레임 워크는 게임 AI 연구에 기여할 수 있길 바란다.

참고문헌

[1] 이만재. “게임에서의 인공지능 기술”, 정보처리학회지, vol.9 no.3 ,pp. 69 - 76 , 2002

[2] 조병헌 외 “게임 인공지능 연구 동향”, 정자통신동향분석, vol.23 no.4 pp.115-121, 2008

[3] 안정근 외 “합성곱 신경망 알고리즘이 도입된 인공지능 인지 시스템을 통한 게임 AI 의 행동 메커니즘 향상 및 FPS 게임의 난이도 조절 방법론 제안”, 디지털콘텐츠학회논문지, vol.23, no.1 pp.21-30, 2022.

[4] 전영진 외 “CNN 기반 기보학습 및 강화학습을 이용한 인공지능 게임 에이전트”, 전기전자학회 논문지, vol.23, no.4 pp.1187-1194, 2019

[5] 조병헌 외 “대전 액션 게임을 위한 신경만 지능 캐릭터의 구현”, 퍼지 및 지능시스템학회 논문지, vol.14, no.4 pp.383 - 389, 2004