강화학습을 통한 체스 게임

위키 팀

파란커피 양지승 최동수 홍지우

목차

팀 소개

안녕하세요, 파란커피 팀입니다!

양지승
@지승

최동수
@토끼

홍지우
@루

- Stanford cs234 강의를 공부하며, '수식 없는 한글 강화학습 위키'를 제작
- 이번 기수에서는 위키 제작과 동시에, '강화학습을 이용한 체스 게임 학습시키기' 프로젝트를 진행

01 강화학습

1. 강화학습

강화학습이란?

불확실성 하에서, **최종적**으로 **최대의 보상**을 받을 수 있도록

연속적으로 이어지는 **일련의 결정**들을 내릴 수 있게 학습시키는 것

1. 강화학습

다른 인공지능 학습법과 비교했을 때, 왜 강화학습인가?

규칙이 이미 정해진 모델이나,

주어진 경험을 토대로 학습하는 다른 모델들과 비교해보았을 때,

새로운 상황에 적용하기 수월하며 성능이 뛰어나다

02 이론적 배경

2. 이론적 배경

시간차 학습

동적 프로그래밍 방법

- World가 **완벽하게 주어졌을 때** 최적의 정책을 계산하기 위해 사용되는 알고리즘
- 학습에 사용할 데이터로 world, model, policy 등 **갖춰야 할 데이터가 많음**
- **정책 평가 -> 정책 향상 -> 정책 반복 or 가치 반복** 의 과정을 통해 **policy를 update**
- **가치 함수**를 활용

  특정 state에서 reward들의 기댓값

2. 이론적 배경

시간차 학습

몬테카를로 방법

- **표본 reward의 평균값**을 기반으로 강화학습 문제를 푸는 방법
- 하나의 에피소드가 **완전히 완료**된 후에만 state에 대한 가치 추정값과 정책이 변화
- 어떤 state 이후 관측되는 모든 reward에 대해 **평균**을 계산하는 것
- 학습에 사용할 데이터로 오로지 **경험**만을 필요로 함

  World와의 상호작용으로부터 방생한
  모든 state, action, reward의 표본을 나열한 것

2. 이론적 배경

시간차 학습

시간차 학습

= 몬테카를로 방법 + 동적 프로그래밍 방법

2. 이론적 배경

시간차 학습과의 유사성

- 몬테카를로 방법

구체적인 환경에 대한 모델 없이도

가공하지 않은 경험으로부터 직접 학습 가능

- DP 방법

동적 프로그래밍 방법

최종 결과를 얻을 때까지 기다리지 않고,

부분적으로는 다른 학습된 추정값을 기반으로 추정값 갱신

2. 이론적 배경

시간차 학습의 장점

DP 방법 사용 -> state에 대한 model, reward, 다음 state의 확률 분포를 모두 필요로 함

몬테카를로 방법 사용 -> 한 에피소드가 완전히 끝날 때 까지 기다려야 함

- 많은 적용 사례에서는 모든 조건을 갖추고 있기 어려움
- 몇몇 적용 사례는 한 에피소드가 굉장히 길기 때문에 **학습이 너무 느리게 진행**
- 어떤 적용 사례는 연속적인 작업이라 **에피소드가 전혀 없음**
- 실험적인 행동이 취해지는 **에피소드를 무시하거나 할인**해야 함

2. 이론적 배경

Q-Learning

Q-Learning

비활성 정책 시간차 학습 제어

2. 이론적 배경

Q-Learning

Q 함수

주어진 **상태(state)**에서 agent가 어떠한 **행동(action)**을 취했을 때

받을 수 있는 **보상(reward)**의 기댓값을 예측하는 함수

2. 이론적 배경

Q-Learning

Q Learning

Q 함수를 사용하여 **최선의 정책(policy)**를 학습하는 강화학습 기법

Model-free 알고리즘으로, agent의 **탐색**을 통해 학습

- 각 state에 대한 action이 어떤 결과를 불러올 지 알 수 없음

2. 이론적 배경

Q-Learning

- Model-free 알고리즘이므로 모델이 주어지지 않을 뿐, **정책은 주어진다**

  -> **정책에 따라 action 결정**, 이후 reward와 state 변화를 관측

- state 별 action에 해당하는 Q-value를 **테이블 형식**으로 **모두 저장**하여 학습

  -> state와 action이 늘어남에 따라 **요구되는 메모리와 탐색 시간 증가**

03 프로젝트

3. 프로젝트

프로젝트 배경

- 그동안 공부한 강화학습을 적용해보기 위한 프로젝트 진행
- 강화학습 프로젝트 중 가장 접근성이 좋고 대표적인 분야는 게임

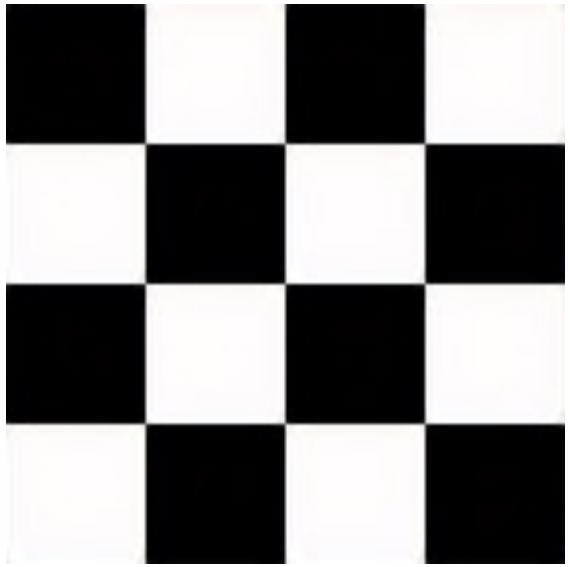
  ⇒ 체스 게임을 강화학습으로 학습시켜보자!

- 모든 말을 사용하는 기본 체스 게임은 이미 많은 프로젝트들이 진행된 바 있음
- 모든 말과 판을 다 사용하면 state가 너무 많아져 프로젝트 볼륨이 너무 커짐

3. 프로젝트

프로젝트 배경

⇒ 4x4 판에서 각 팀이 폰 4개씩만 사용하는 게임으로 만들어보자!



3. 프로젝트

프로젝트 진행

1. 선행 연구 조사 - AlphaGo Zero

AlphaGo Zero 방식이란?

1. 컴퓨터가 스스로 플레이하며 모든 동작들을 기록, 말의 모든 이동들을 기록하여 주어진 체스 페이스가 승리인지, 패배인지 판단하는 방법을 학습
2. 1의 신경망을 복제하고, 이전 state에서 얻었던 데이터 세트로 복제된 신경망을 훈련시킴
3. 1의 신경망과 2에서 복제된 신경망끼리 게임 플레이
4. 둘 중 패배한 신경망은 버리고, 승리한 신경망을 이용하여 2-3단계를 반복
5. 1-4단계 반복

3. 프로젝트

프로젝트 진행

2. 실험 계획

- 체스 게임을 구현할 수 있는 기존 라이브러리가 존재하나, 이번 프로젝트에서 사용할 게임 방법인 pawn chess에 적용할 수 없음 -> **새로운 게임 엔진 구현** 필요
- 폰이 선택할 수 있는 action은 **전진, 좌측 공격, 우측 공격**의 3가지가 있음
- 기물 종류가 1개(폰) 뿐이기 때문에, **각 폰의 넘버로만 구별함 (W1 ,W2, W3, W4)**
- **state**, **action**, **value** 값으로 **Q-table**을 만들어 여러 게임을 거치며 Q-table update 진행

3. 프로젝트

프로젝트 진행

3. 데모 및 게임 엔진 구현

	0	1	2	3
0	B	B	B	B
1	-	-	-	-
2	-	-	-	-
3	W	W	W	W

▲ 시작 배치

	0	1	2	3
0	B	-	B	B
1	-	B	-	-
2	-	W	W	-
3	W	-	-	W

▲ 게임 진행

- 대강의 게임 과정을 그려보기 위해, 게임 엔진의 데모 버전을 제작
- 폰의 넘버는 action 선택에 영향을 미치지 않음

  -> 폰의 색깔만 구분하고, 폰의 넘버는 구별하지 않기로 함
- Q-table의 state는 판의 모양 그대로 저장하는 것이 아닌,

  'B-BB-B---WW-W—W'와 같이 저장

3. 프로젝트

프로젝트 진행

3. 데모 및 게임 엔진 구현

- 게임 진행

  1. 4 x 4 판에서 (0~3)*(0~3)사이 무작위로 좌표를 선정
  2. 그 좌표에서 다시 무작위로 0 = 전진, 1=좌측공격, 2=우측공격 중 하나의 action 선택
  3. 만약에 불가능한 움직임이면 다시 좌표선택부터 반복
  4. 최종적으로 얼마나 더 많은 기물이 남았는가에 따라 승리, 무승부, 패배 측정

- 결과

  선공인 백측의 승률이 33.26%, 후공인 흑측의 승률이 31.28%, 무승부 35.46%

  => 33%에 근접한 세 개의 결과값

3. 프로젝트

프로젝트 진행

4. 학습 준비

- 규칙

1. 폰은 전진, 좌측 공격, 우측 공격의 3가지 action 중 하나를 선택할 수 있음

2. state-action의 데이터를 쌓아 Q-table 값을 얻기 위해 초기에는 무작위로 움직임

3. 한 경기를 하여서 이겼을 때는 0.9, 무승부 시 0.6, 패배 시 0.3을 개별 Q-table의 reward로 부여

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	BBBB--------WWWW	0	0	0	0	1	0	0	0	0	0	0	0
1	-BBBB----W--W-WW	0	0	0	0	0	0	0	0	0	0	1	0
2	--BBBB---W-WW-W-	0	0	0	1	0	0	0	0	0	0	0	0
3	--BBW----B-WW-W-	0	1	0	0	0	0	0	0	0	0	0	0
4	--BBW---W--W-BW-	0	0	0	0	1	0	0	0	0	0	0	0
5	W--B--B-W--W-BW-	0	0	0	0	0	0	0	1	0	0	0	0
6	W-----BBW-WW-B--	0	1	0	0	0	0	0	0	0	0	0	0
7	W---W-B---BW-B--	0	0	0	0	0	0	0	0	0	1	0	0
8	W---W-W------BB-	0	0	0	0	0	0	0	0	0	0	1	0
9	BBBB--------WWWW	0	0	0	0	0	0	0	1	0	0	0	0
10	B-BB-B----W-WW-W	0	1	0	0	0	0	0	0	0	0	0	0
11	B-B--B-BW-W--W-W	0	0	0	0	1	0	0	0	0	0	0	0
12	B-B--B--WWWB---W	0	1	0	0	0	0	0	0	0	0	0	0
13	B---WBB--WWB---W	0	0	0	0	0	0	1	0	0	0	0	0
14	----WBB--W-B---W	0	0	0	0	0	1	0	0	0	0	0	0
15	----W-W--B-B---W	0	0	0	0	1	0	0	0	0	0	0	0
16	--W-W------B-B-W	0	1	0	0	0	0	0	0	0	0	0	0
17	BBBB--------WWWW	0	0	0	0	0	0	0	0	0	0	1	0
18	BB-B--B----WWWW-	0	1	0	0	0	0	0	0	0	0	0	0
19	BB----BBW--W-WW-	0	1	0	0	0	0	0	0	0	0	0	0
20	B---B-BB---W-WW-	0	0	0	0	1	0	0	0	0	0	0	0
21	B-----BBBW-W--W-	0	0	0	0	0	0	0	1	0	0	0	0
22	B------BBWWB----	0	0	0	0	0	0	0	1	0	0	0	0
23	----B-WBBW-B----	0	0	0	0	0	0	0	1	0	0	0	0

3. 프로젝트

프로젝트 진행

5. 초기 Q-table 구현

- state-action 별 value 값을 나타냄
- 각 state 별 action들 중 어떤 action이 가장 높은 value 값을 가지는지 확인

  -> 가장 높은 value값을 가지는 action이 가장 적합한 action
- 앞서 나온 Q-table 값의 평균을 도출

action

state

	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
---------B-W----	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.24	0.00	0.0	0.36	0.0
---------BWB----	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.28	0.00	0.0	0.00	0.0
---------W-B----	0.0	0.000000	0.0	0.0	0.4	0.0	0.0	0.20	0.00	0.0	0.00	0.0
---------WBW----	0.0	0.920000	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.00	0.0
--------B-BW----	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.28	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
WW----B-------BW	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.76	0.0
WW----B----B--WW	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.00	0.94	0.0	0.00	0.0
WW----B----BW--W	0.0	0.946667	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.00	0.0
WW----W------B-B	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.92	0.0
WW-B---------BBW	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.60	0.0

15764 rows × 12 columns

value

3. 프로젝트

프로젝트 진행

6. 1차 학습

- **규칙**

1. 폰은 전진, 좌측 공격, 우측 공격의 3가지 action 중 하나를 선택할 수 있음

2. 흑과 백 중 한 쪽이라도 더 이상 움직일 수 없는 상태가 되면 게임 종료

3. 초기 Q-table 값에 기반하여 움직임

4. 패배했을 때의 reward를 기존 값의 0.9, 승리했을 때의 reward를 1.1, 무승부일 때는 동일하게 업데이트

5. 기존에 없던 state 등장 시 초기 Q-table 값과 동일 기준 적용

3. 프로젝트

프로젝트 진행

6. 1차 학습

- **결과**
  1. 업데이트가 진행되었지만 무조건적으로 Q-table에만 기반한 action을 취했을 때 특정한 경로만 학습

  2. 최적 action에 대한 결과가 무승부일 때, 업데이트를 따로 해주지 않으므로 두 agent들이 일정 게임 이후부터는 업데이트 되지 않음.

3. 프로젝트

프로젝트 진행

7. 2차 학습

- **Reward 할당 비율 변경**

  1. 패배했을 때의 reward* = reward x (10/11+score*0.1), 승리했을 때의 reward* = reward x (11/10+score*0.1), 무승부 일 때의 reward* = reward x (1+score*0.1)으로 설정

  2. score는 종료 시 남아있는 기물의 숫자, 전진한 정도에 따른 점수

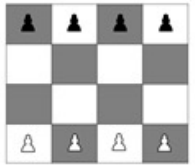
  3. experience replay buffer 문제를 해결하기 위해서 일정 확률로 Q-table 값을 따르지 않고 다른 action을 취하는 방식으로 업데이트

3. 프로젝트

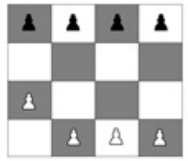
프로젝트 진행

7. 2차 학습

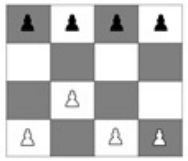
- 가장 왼쪽에 state가 위치
- 위의 1-L, ...4-R은 action을 의미
- 각 state-action 쌍 별 value값이 존재
- Agent는 저 value값 중 가장 높은 값을 가지는 action을 수행함



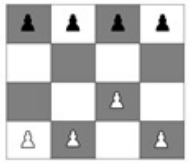
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.038742	0.0	0.0	0.078822	0.0	0.0	0.064007	0.0	0.0	0.313205	0.0



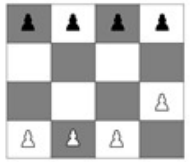
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
-BBBB------WWWW-	0.0	0.156913	0.0	0.0	0.15487	0.0	0.0	0.183642	0.0	0.0	0.186939	0.0



	1-L	1-D	1-R	2-L	2-D	2-R	3-L	3-D	3-R	4-L	4-D	4-R
BBBB------W-WW-W	0.0	0.098157	0.0	0.0	0.186695	0.0	0.0	0.723167	0.0	0.0	0.087569	0.0



	1-L	1-D	1-R	2-L	2-D	2-R	3-L	3-D	3-R	4-L	4-D	4-R
BBBB-----W--W-WW	0.0	0.34418	0.0	0.0	0.215727	0.0	0.0	0.110412	0.0	0.0	0.114137	0.0



	1-L	1-D	1-R	2-L	2-D	2-R	3-L	3-D	3-R	4-L	4-D	4-R
BBBB----W----WWW	0.0	0.258145	0.0	0.0	0.272363	0.0	0.0	0.130837	0.0	0.0	0.210378	0.0

3. 프로젝트

프로젝트 진행

7. 2차 학습

4-U의 값을 확인해보면 게임을 2번 수행한

이후의 값은 낮지만, 100번의 게임을

수행한 이후의 값이 높아진 것을 볼 수 있음

```
# First Point
pd.DataFrame ( return_row(w_mean, 'BBBB--------WWWW')).T
```

	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.150188	0.0	0.0	0.151232	0.0	0.0	0.15198	0.0	0.0	0.126173	0.0

```
# After 1 Game
pd.DataFrame ( return_row(w_mean, 'BBBB--------WWWW')).T
```

	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.150188	0.0	0.0	0.151232	0.0	0.0	0.15198	0.0	0.0	0.123578	0.0

```
# After 2 Game
pd.DataFrame ( return_row(w_mean, 'BBBB--------WWWW')).T
```

	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.150218	0.0	0.0	0.151262	0.0	0.0	0.15201	0.0	0.0	0.121036	0.0

```
# After 100 Game
pd.DataFrame ( return_row(w_mean, 'BBBB--------WWWW')).T
```

	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.15076	0.0	0.0	0.151808	0.0	0.0	0.152559	0.0	0.0	0.152681	0.0

3. 프로젝트

프로젝트 진행

8. 3차 학습

초기 Q-table 값 없이 진행

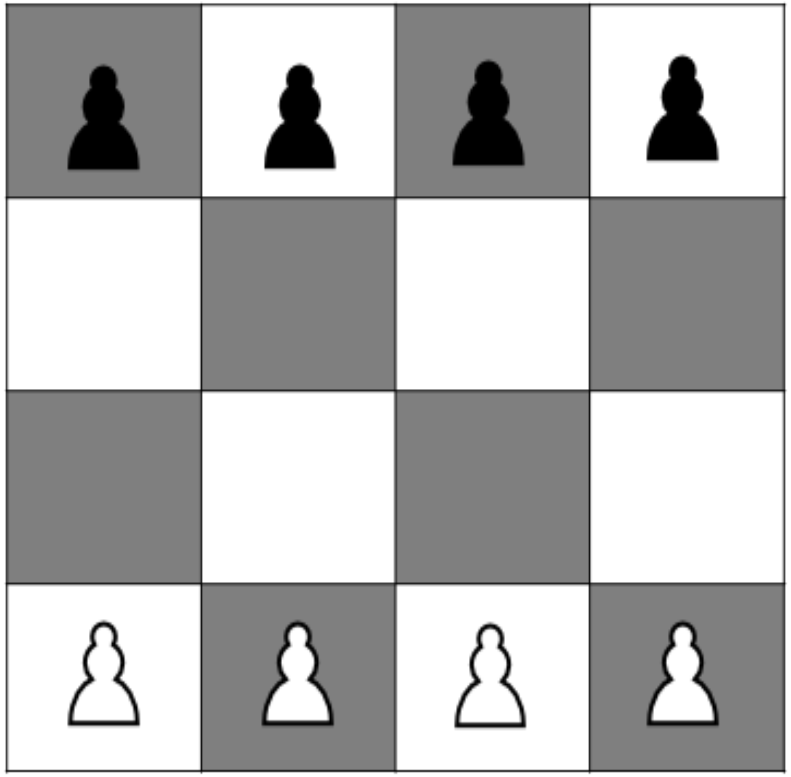
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	1.0106	0.0
B-BB-B-----WWWW-	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	1.0106	0.0	0.0	0.0000	0.0
B-BB-----BWWWW--	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	1.0106	0.0	0.0	0.0000	0.0
--BBB-W--B-WWW--	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	1.0106	0.0
--BB--WWBB--WW--	0.0	0.0000	1.0106	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
---B--WBBW---W--	0.0	0.0000	0.0000	1.0106	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
------BBWW------	0.0	0.0000	1.0106	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
------W-W--B----	0.0	1.0106	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
--W-----W------B	0.0	0.0000	0.0000	0.0000	1.0106	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0

	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.0000	0.0000	0.0000	1.0002	0.0	0.0	0.0000	0.0	0.0	1.0106	0.0
B-BB-B-----WWWW-	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	1.0106	0.0	0.0	0.0000	0.0
B-BB-----BWWWW--	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	1.0106	0.0	0.0	0.0000	0.0
--BBB-W--B-WWW--	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	1.0106	0.0
--BB--WWBB--WW--	0.0	0.0000	1.0106	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
---B--WBBW---W--	0.0	0.0000	0.0000	1.0106	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
------BBWW------	0.0	0.0000	1.0106	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
------W-W--B----	0.0	1.0106	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
--W-----W------B	0.0	0.0000	0.0000	0.0000	1.0106	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
-BBBB----W--W-WW	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	1.0002	0.0	0.0	0.0000	0.0
-B-BB-B--WW-W--W	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	1.0002	0.0
-B-B--B--BWWW---	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	1.0002	0.0
-B-B--BW--W-WB--	0.0	1.0002	0.0000	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0
---B-BBWW-W--B--	0.0	0.0000	1.0002	0.0000	0.0000	0.0	0.0	0.0000	0.0	0.0	0.0000	0.0

3. 프로젝트

프로젝트 결과

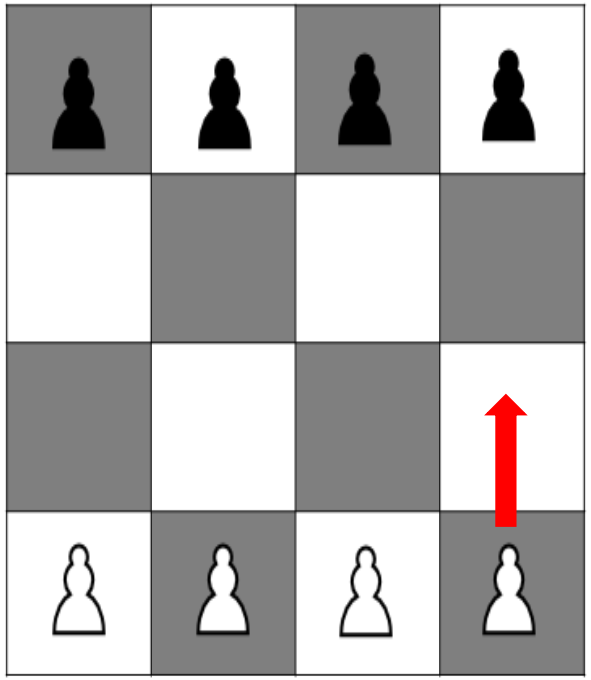
9. 게임 예시



3. 프로젝트

프로젝트 결과

9. 게임 예시 : AI턴



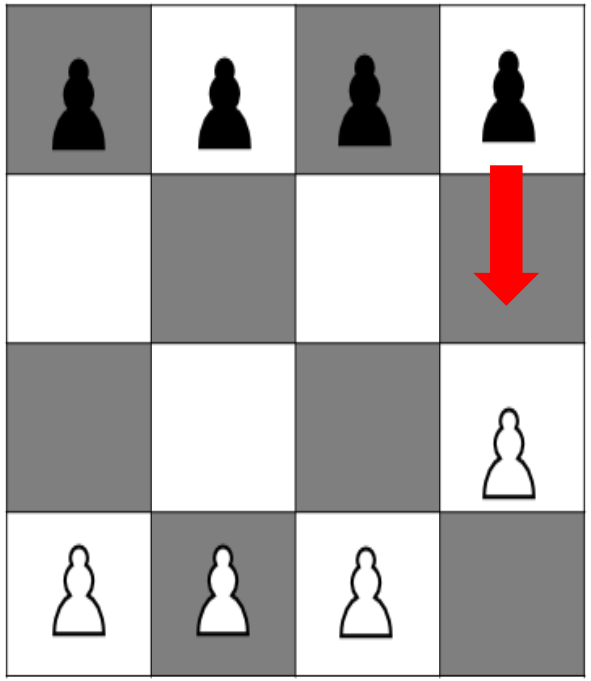
	0	1	2	3
0	B1	B2	B3	B4
1	--	--	--	--
2	--	--	--	--
3	W1	W2	W3	W4

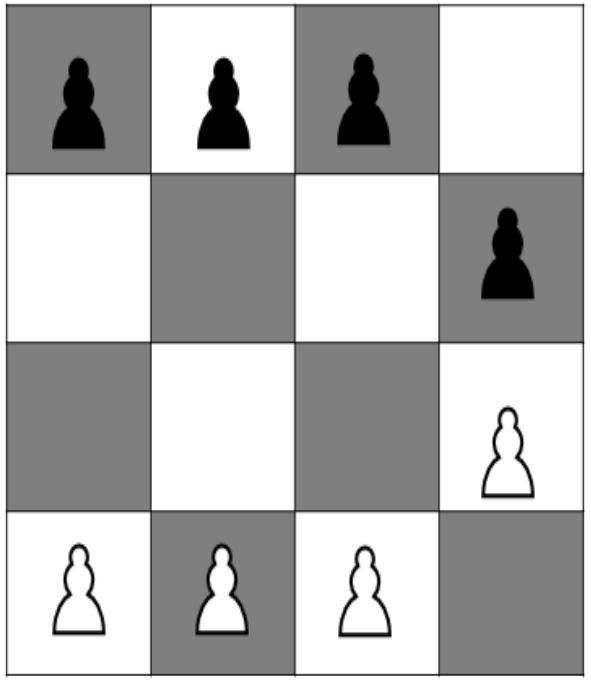
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBBB--------WWWW	0.0	0.038742	0.0	0.0	0.078822	0.0	0.0	0.064007	0.0	0.0	0.313205	0.0

3. 프로젝트

프로젝트 결과

9. 게임 예시 : 상대 턴

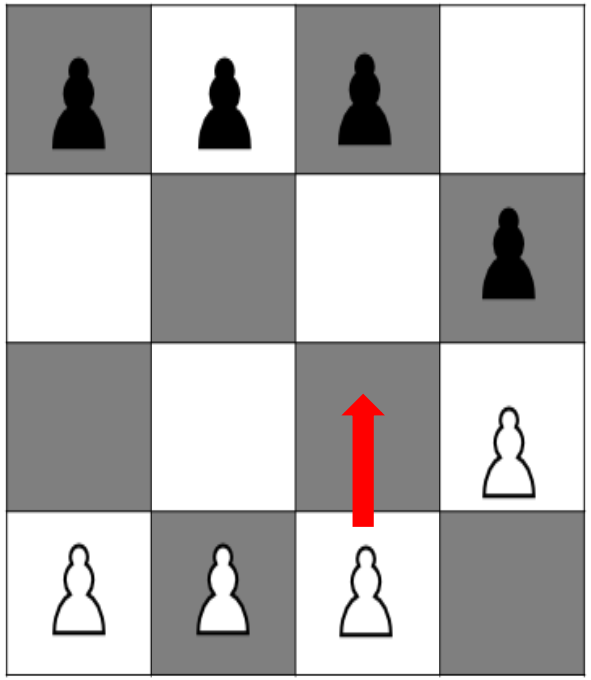




3. 프로젝트

프로젝트 결과

9. 게임 예시 : AI턴



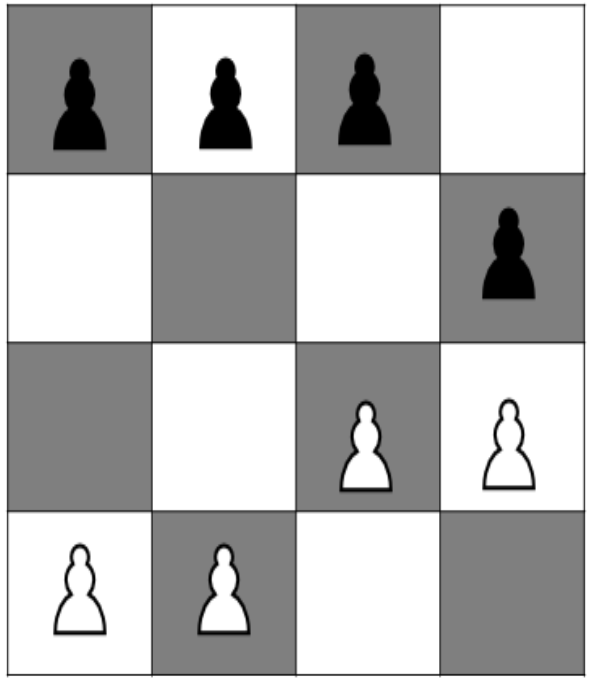
	0	1	2	3
0	B1	B2	B3	--
1	--	--	--	B4
2	--	--	--	W4
3	W1	W2	W3	--

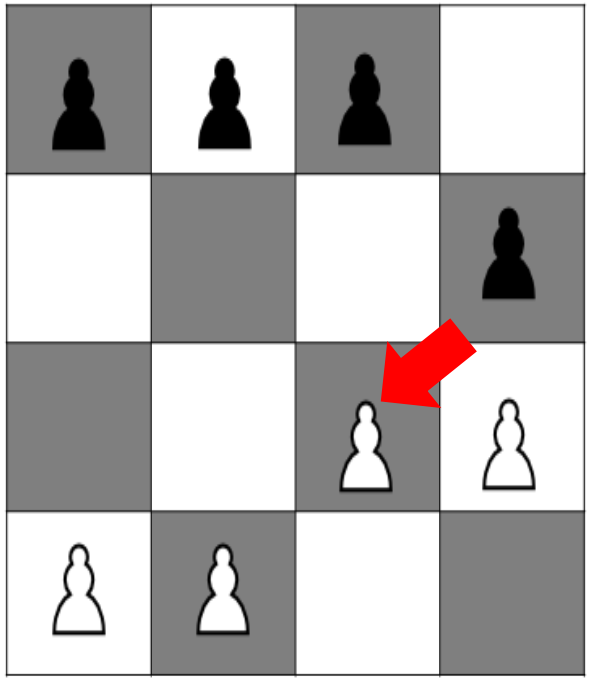
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	-R	4-L	4-U	4-R
BBB----B---WWWW-	0.0	0.562844	0.0	0.0	0.450288	0.0	0.0	0.864261	0.0	0.0	0.0	0.0

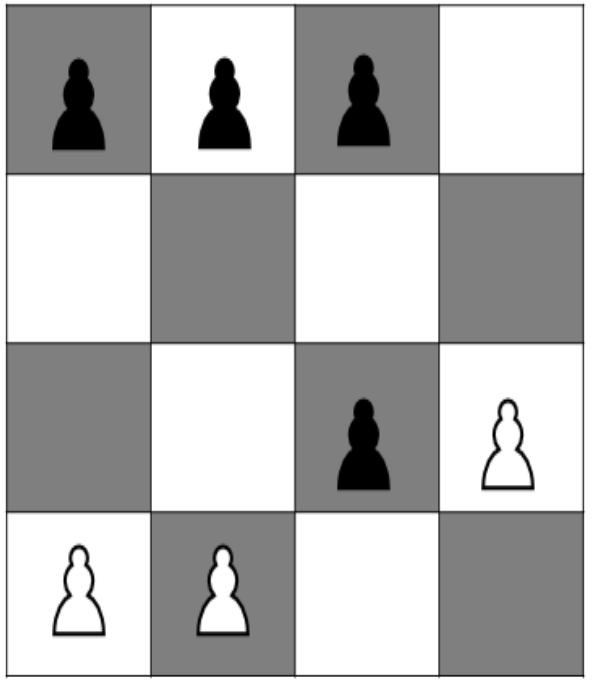
3. 프로젝트

프로젝트 결과

9. 게임 예시 : 상대 턴



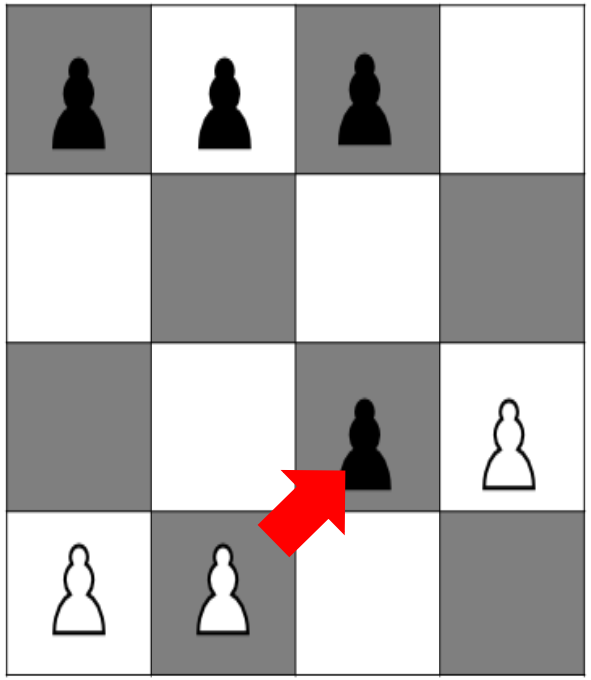




3. 프로젝트

프로젝트 결과

9. 게임 예시 : AI턴



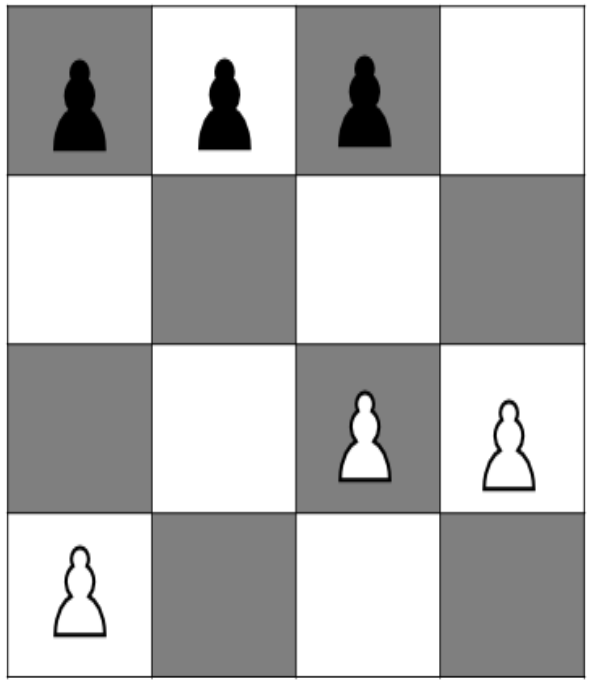
	0	1	2	3
0	B1	B2	B3	--
1	--	--	--	--
2	--	--	B4	W4
3	W1	W2	--	--

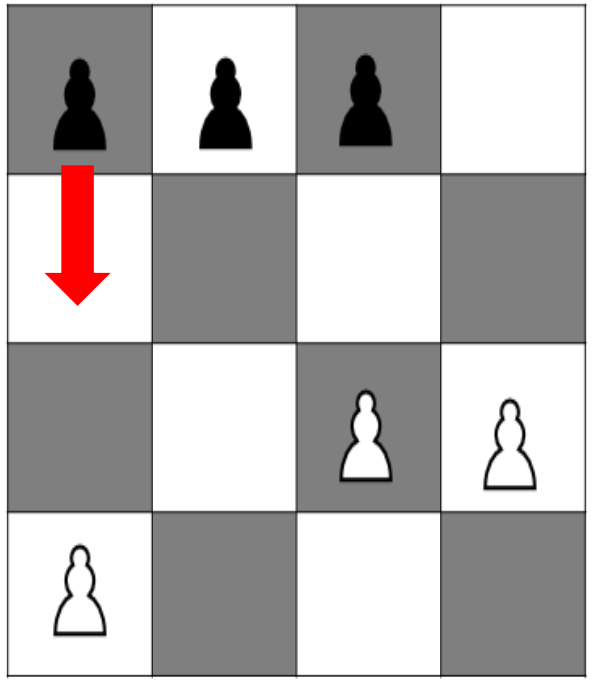
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
BBB-------BWWW--	0.0	0.083036	0.0	0.0	0.081531	0.114989	0.0	0.0	0.0	0.0	0.090319	0.0

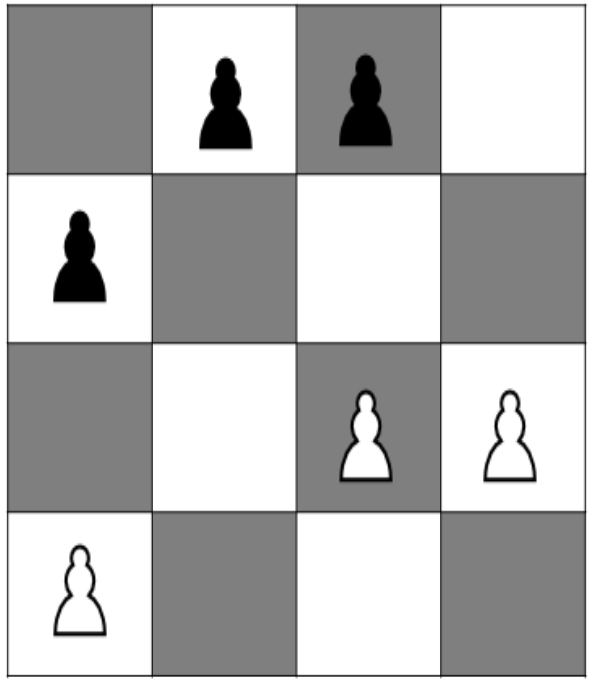
3. 프로젝트

프로젝트 결과

9. 게임 예시 : 상대 턴



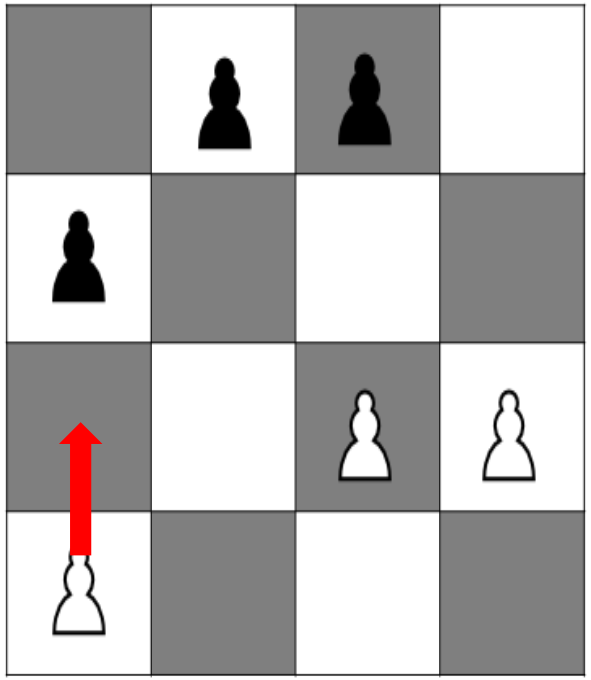




3. 프로젝트

프로젝트 결과

9. 게임 예시 : AI턴



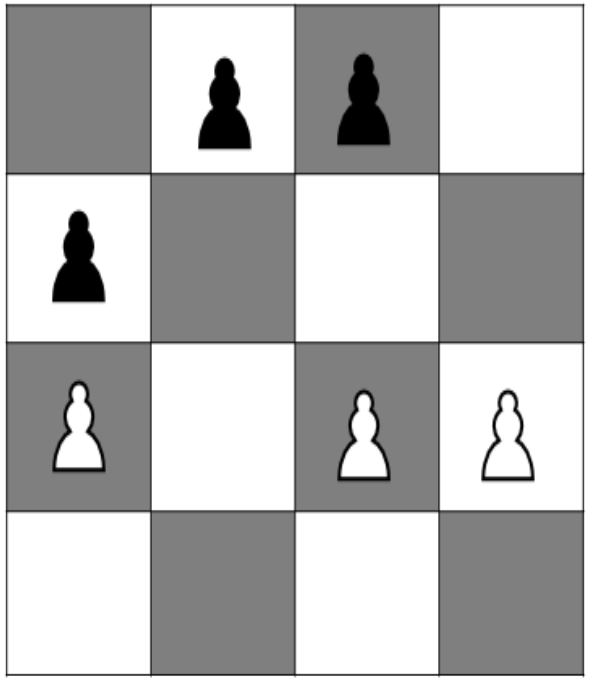
	0	1	2	3
0	--	B2	B3	--
1	B1	--	--	--
2	--	--	W2	W4
3	W1	--	--	--

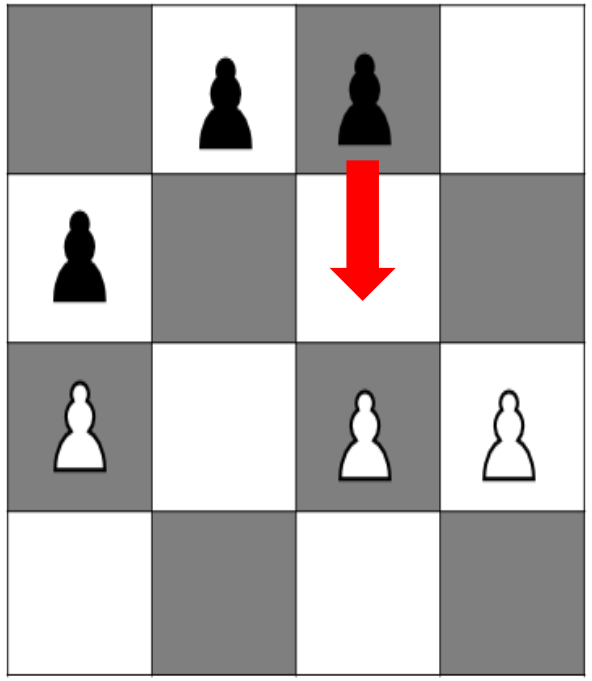
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
-BB-B-----WWW---	0.0	0.275362	0.0	0.0	0.147036	0.0	0.0	0.0	0.0	0.0	0.252174	0.0

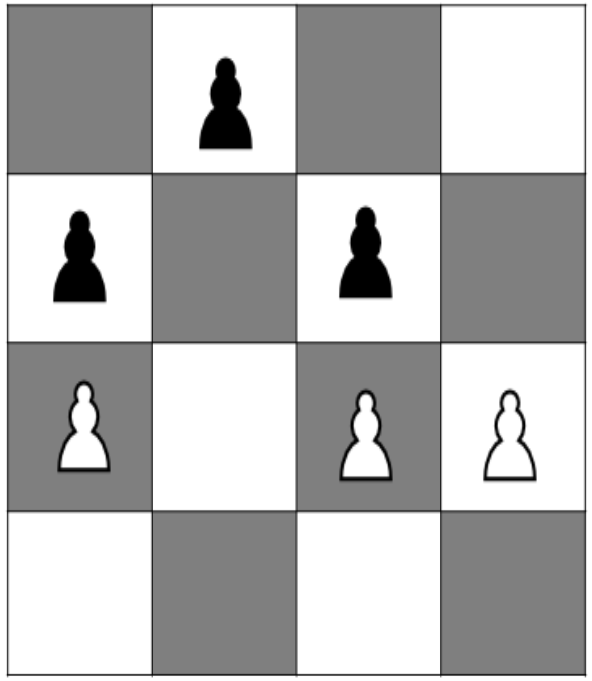
3. 프로젝트

프로젝트 결과

9. 게임 예시 : 상대 턴



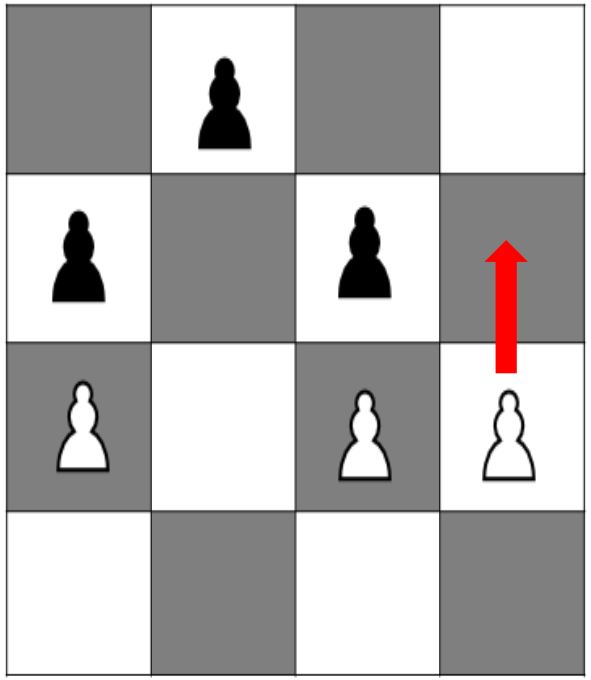




3. 프로젝트

프로젝트 결과

9. 게임 예시 : AI턴



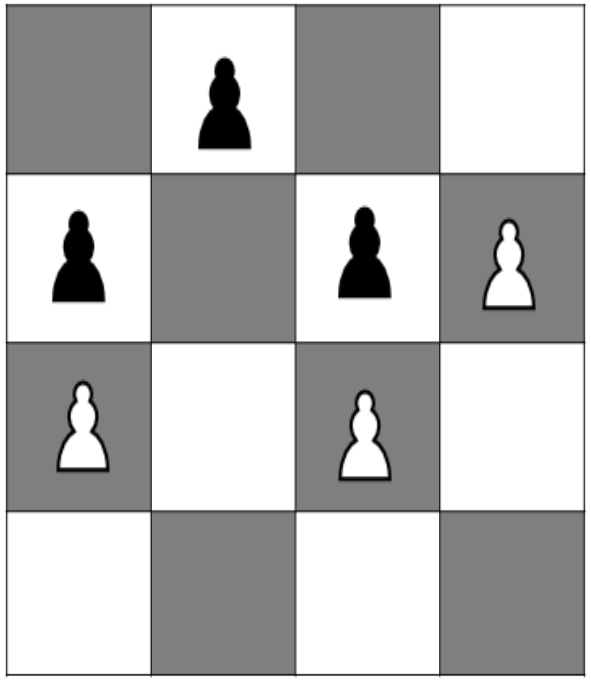
	0	1	2	3
0	--	B2	--	--
1	B1	--	B3	--
2	W1	--	W2	W4
3	--	--	--	--

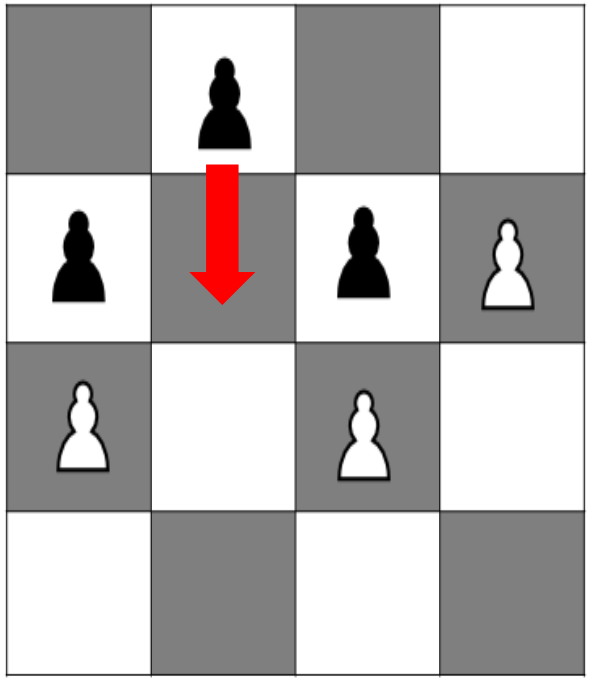
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
-B--B-B-W-WW----	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.35336	0.443146	0.0

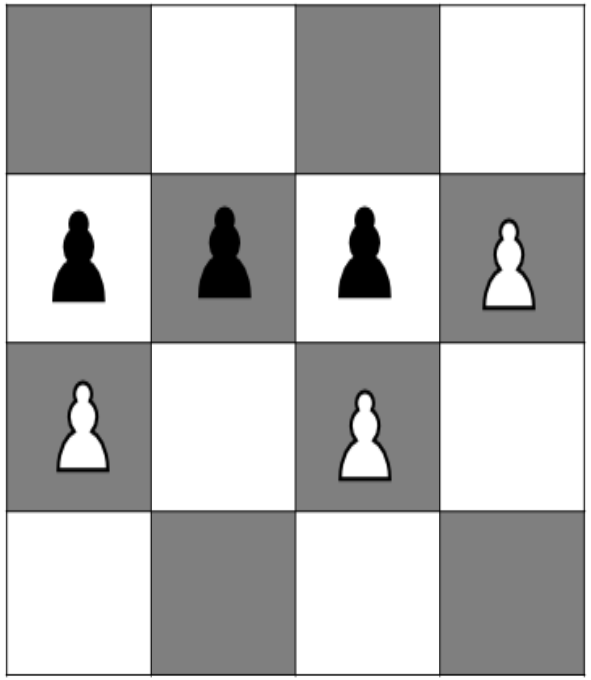
3. 프로젝트

프로젝트 결과

9. 게임 예시 : 상대 턴



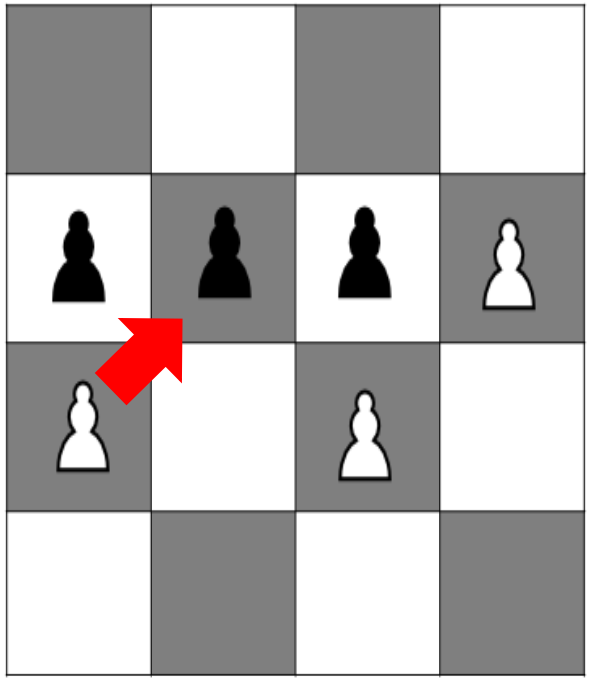




3. 프로젝트

프로젝트 결과

9. 게임 예시 : AI턴



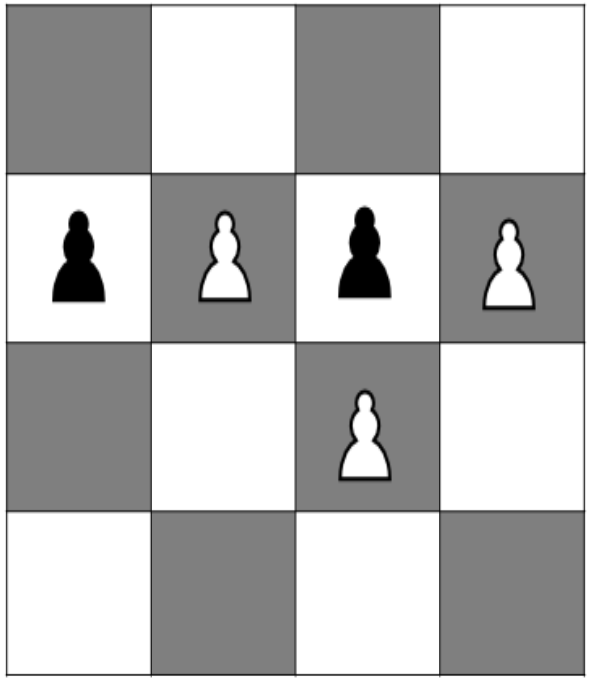
	0	1	2	3
0	--	--	--	--
1	B1	B2	B3	W4
2	W1	--	W2	--
3	--	--	--	--

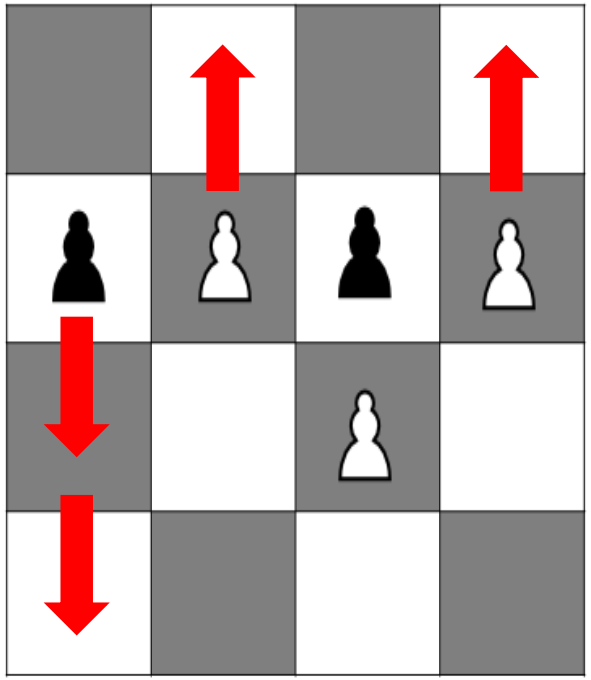
	1-L	1-U	1-R	2-L	2-U	2-R	3-L	3-U	3-R	4-L	4-U	4-R
----BBBWW-W-----	0.0	0.0	0.386158	0.314737	0.0	0.0	0.0	0.0	0.0	0.0	0.104737	0.0

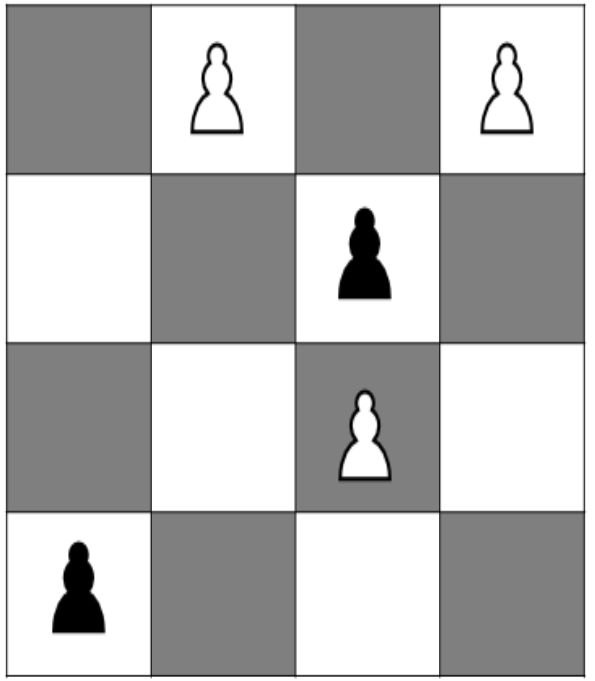
3. 프로젝트

프로젝트 결과

9. 게임 예시 : 마무리







3. 프로젝트

프로젝트 발전 방향

프로젝트 발전 방향

- 이후 판의 크기를 키우거나(8x8, 16x16), 체스 말을 늘려 적용
- 말을 W,B로 표시하는 것이 아닌 실제 체스판처럼 시각화
- Q-Learning 외에 DQN 등의 방법으로 학습시켜 성능 비교

3. 프로젝트

프로젝트 데모 시연

발표가 끝난 후 점심식사&데모&네트워킹 행사 시간에
파란커피 팀이 학습시킨 모델과 체스 게임을 직접 해볼 수 있습니다!

Q&A

감사합니다