Seoul AI Gym

Martin Kersner

Seoul AI Gym

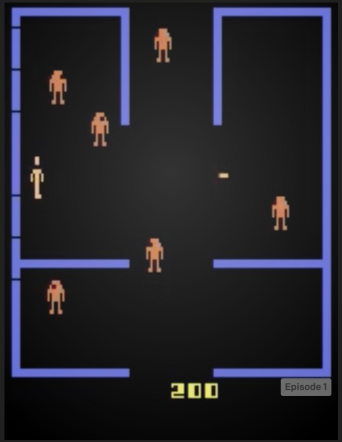
- Seoul AI Gym is a toolkit for developing AI algorithms
- Similar API to Open AI Gym (https://gym.openai.com/)
- Currently, support game of Checkers
- Python 3.6

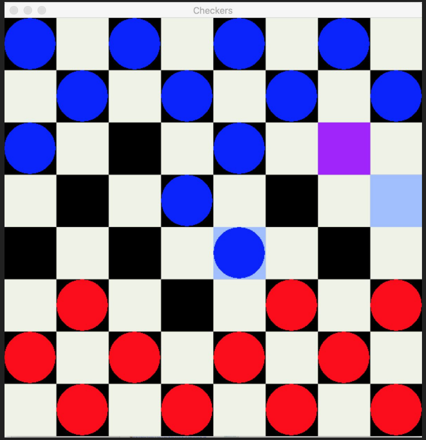
https://github.com/seoulai/gym

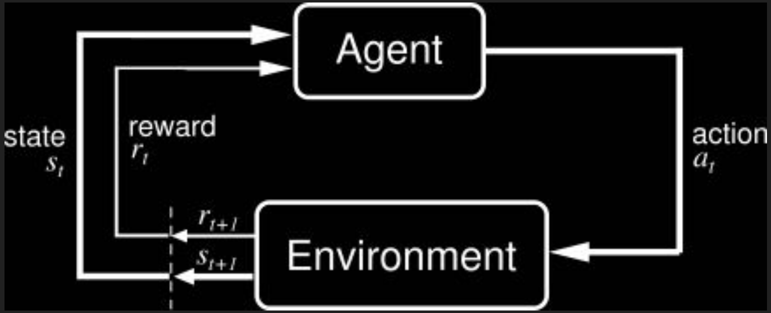
Environment and Agent

An **Environment** is a world (= simulation) with which an **Agent** can interact.

An **Agent** can observe a world and act based on its decision.







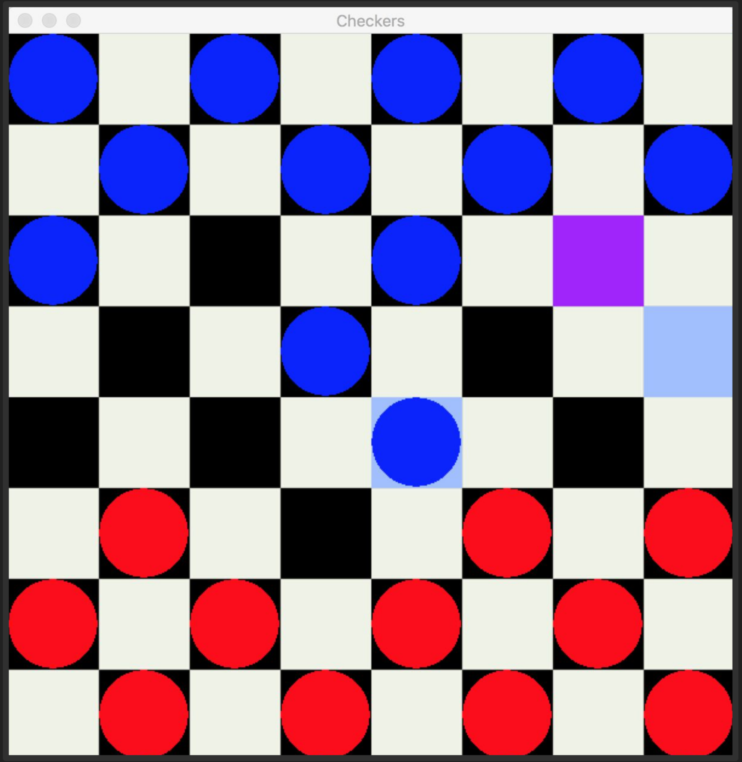
Why to make another gym?

- Open AI Gym does not offer any Environment where Agents could compete with each other.
- Common topic for the next Seoul AI Hackathon: Checkers.

Checkers

Rules

- 8x8 play field
- 2 players
- Light and dark pieces
- All start on black squares
- Move only diagonally forward
- Step size 1 or 2 (when killing)
- Kill (= jump over opponent's piece)
- Become king when reach the end
- King can move forward and backward
- Win when
  - Opponent lost all pieces
  - Opponent can't move



How to start?

Installation

Pip

```
pip3 install seoulai-gym
```

From source

```
git clone https://github.com/seoulai/gym.git
cd gym
pip3 install -e .
```

Checkers

Initialization

```
import seoulai_gym as gym
env = gym.make("Checkers")
obs = env.reset()
env.render()
env.close()
```

https://github.com/seoulai/gym/blob/master/seoulai_gym/envs/checkers/checkers.py

Example game loop

https://github.com/seoulai/gym/blob/master/examples/checkers_example.py

Checkers

Agent

```
class RandomAgent(Agent):
  def __init__(self,
      name: str,
      ptype: int,
  ):
    super().__init__(name, ptype)

  def act(self,
    board: List[List],
    reward: int, # FIXME float
    done: bool,
  ) -> Tuple[int, int, int, int]:
    # decide where to move one of your pieces
```

https://github.com/seoulai/gym/blob/master/seoulai_gym/envs/checkers/agents.py

Random agents demo

Checkers

Auxiliary functions I

```
board_list2numpy(List[List[Piece]]) -> np.array

array([[2., 0., 2., 0., 2., 0., 2., 0.],
       [0., 2., 0., 2., 0., 2., 0., 2.],
       [2., 0., 2., 0., 2., 0., 2., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 1., 0., 1., 0., 1.],
       [1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 1., 0., 1., 0., 1.]])
```

https://github.com/seoulai/gym/blob/master/seoulai_gym/envs/checkers/base.py

Checkers

Auxiliary functions II

```
get_opponent_type(ptype: int) -> int

get_positions(
  board_list: List[List[Piece]],
  ptype: int,
  board_size: int) -> List[Tuple[int, int]]

get_valid_moves(
  board_list: List[List[Piece]],
  from_row: int,
  from_col: int) -> List[Tuple[int, int]]
```

https://github.com/seoulai/gym/blob/master/seoulai_gym/envs/checkers/rules.py

Checkers

Auxiliary functions III

```
generate_valid_moves(
  board_list: List[List[Piece]],
  ptype: int,
  board_size: int) -> Dict[Tuple[int, int], List[Tuple[int, int]]]

validate_move(
  board_list: List[List[Piece]],
  from_row: int,
  from_col: int,
  to_row: int,
  to_col: int) -> bool
```

https://github.com/seoulai/gym/blob/master/seoulai_gym/envs/checkers/rules.py

Checkers

Auxiliary functions IV

```
get_between_position(
  from_row: int,
  from_col: int,
  to_row: int,
  to_col: int) -> Tuple[Optional[int], Optional[int]]

generate_all_moves(
  from_row: int,
  from_col: int) -> List[Tuple[int, int]]
```

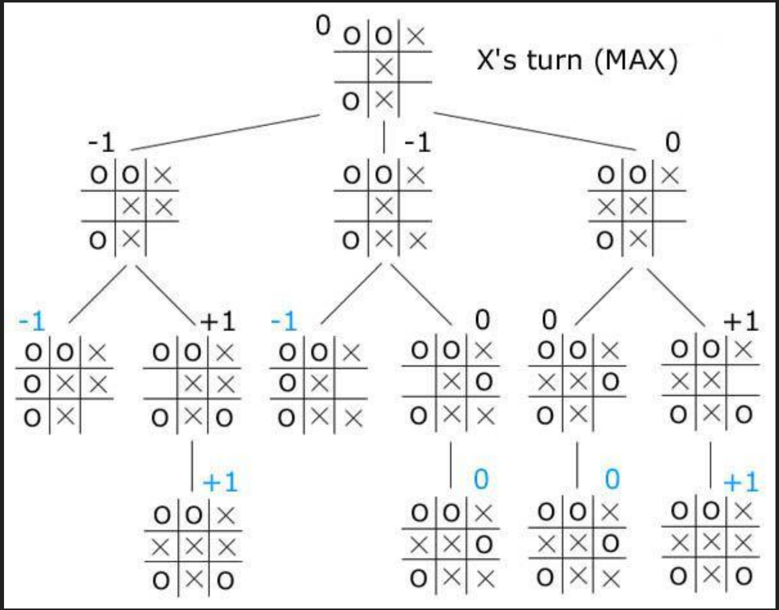
https://github.com/seoulai/gym/blob/master/seoulai_gym/envs/checkers/rules.py

Checkers

Possible solutions I

Minimax

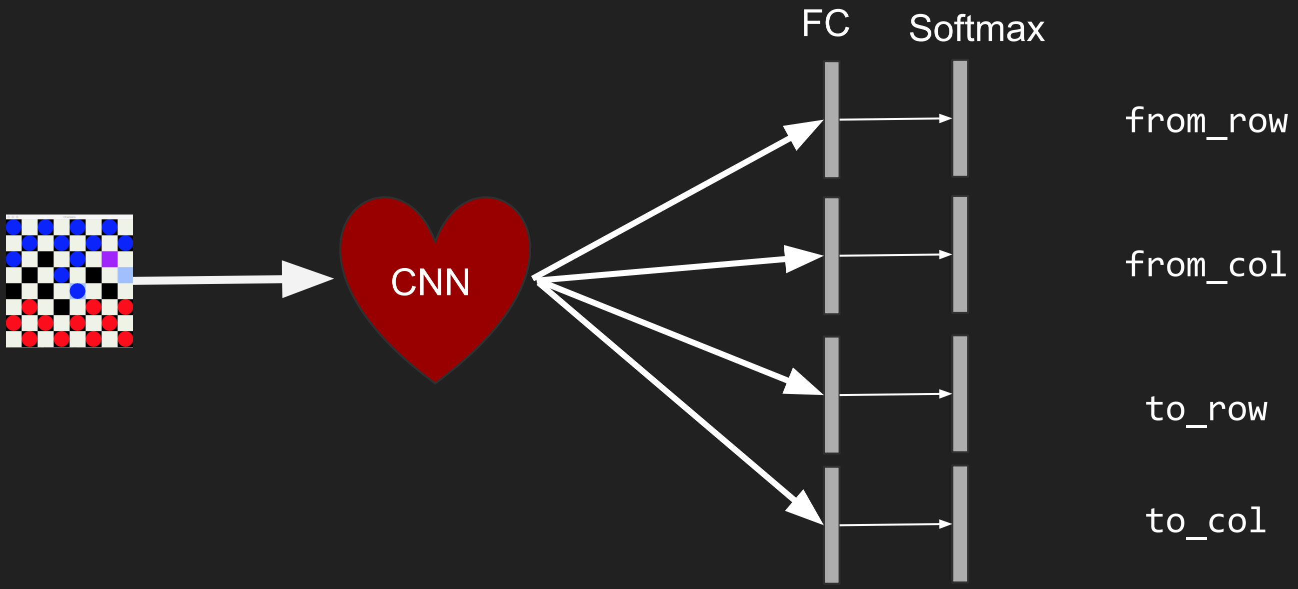
https://en.wikipedia.org/wiki/Minimax



Checkers

Possible solutions II

Deep Q Learning https://en.wikipedia.org/wiki/Q-learning



Seoul AI Gym

- Looking for contributors
- Agents
- Environments
- Code quality
- …
- **Hackathon** (end of summer)