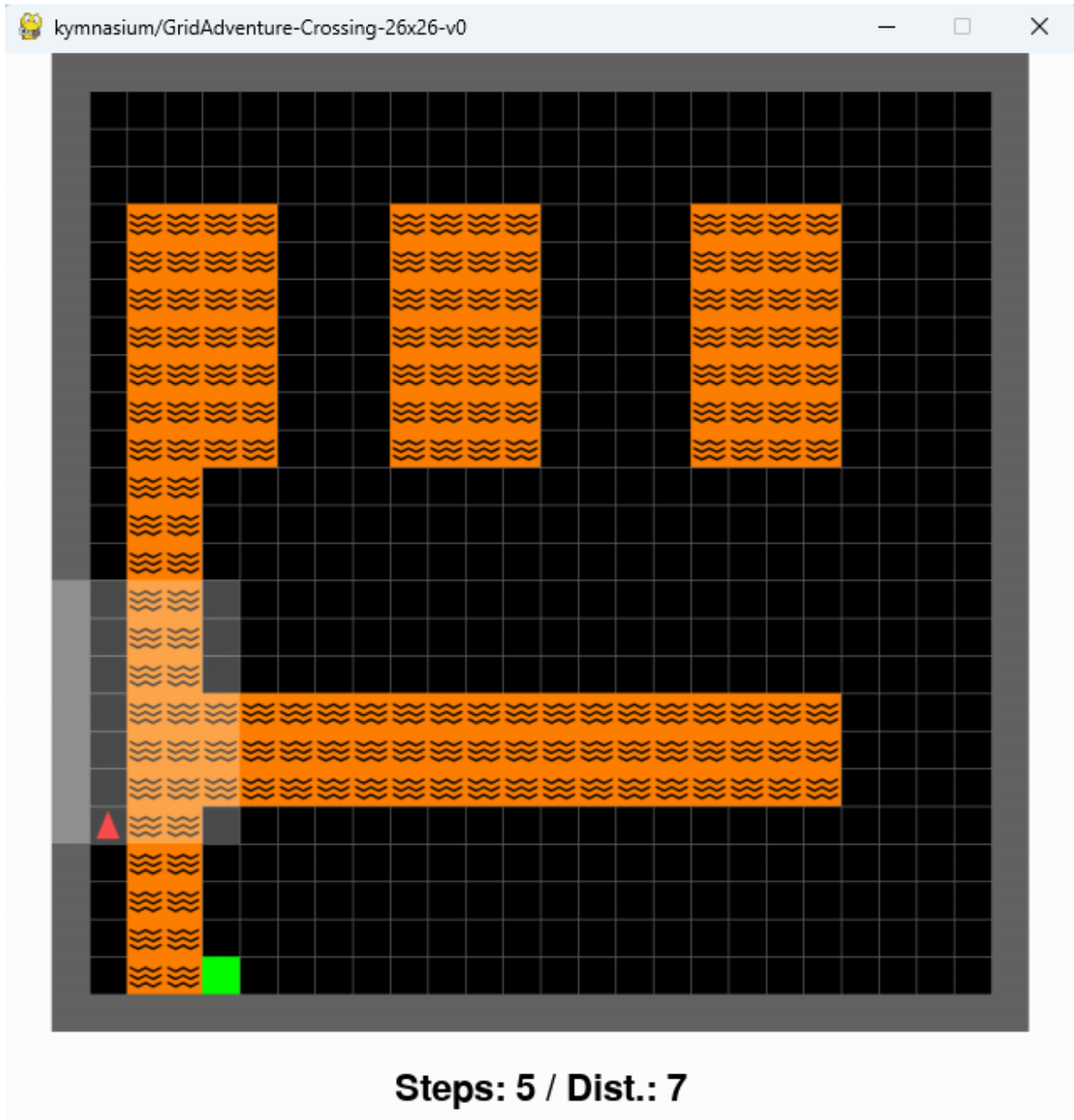


2025-02 KNU RL Competition Round 1

Grid Crossing!



첫 번째 RL Competition은 뜨거운 용암을 피해서 목표 지점으로 향하는 *Grid Crossing*입니다!

Purpose

플레이어는 출발 지점에서 녹색 사각형으로 표시된 도착 지점으로 가야만 합니다. 하지만, 그 앞에는 용암이 가득합니다. 여러분은 용암을 피해서 미로를 최대한 빨리 탈출하는 에이전트를 학습시켜야 합니다.

Prerequisites

Python 3.12.x

Grid Crossing은 Python 3.12 버전에서 동작하도록 구현되었습니다 (그 외의 버전에서는 동작을 보장할 수 없습니다)

Installation

Grid Crossing과 그 외 과제에서 수행할 강화 학습 환경은 다음과 같이 설치할 수 있습니다:

```
pip install -U kymnasium
```

그 외 실행에 필요한 `gymnasium`, `pygame`, `numpy` 등의 라이브러리는 의존성으로 함께 설치되므로 별도 설치할 필요는 없습니다.

Methods and Class

Common Methods

Grid Crossing, 그리고 앞으로 여러분들이 과제로 수행할 `kymnasium` 환경에서 구현에 신경써야 할 클래스 및 함수는 다음과 같습니다.

`kymnasium.Agent`

`kymnasium` 환경을 플레이하는 에이전트를 위한 추상 클래스입니다. 이 추상 클래스를 상속해서 여러분만의 에이전트를 정의하셔야 합니다. 이 추상 클래스에서 여러분이 구현하셔야 할 함수는 **`save`**, **`load`**, **`act`**입니다.

```
import kymnasium as kym
from typing import Dict, Any
```

```
class YourAgent(kym.Agent):
    def save(self, path: str):
        pass

    @classmethod
    def load(cls, path: str) -> 'kym.Agent':
        pass

    def act(self, observation: Any, info: Dict) -> Any:
        pass
```

- **save(path: str):** 훈련시킨 강화학습 에이전트를 path 경로에 파일로 저장하는 함수로, Python의 기본 직렬화 라이브러리인 pickle 등을 활용 가능합니다 ( 6. Tutorial on Gymnasium.ipynb 를 참조하세요).
- **load(path: str):** path 경로에 저장된 에이전트 파일을 불러오는 함수 (예. `YourAgent.load(path)`)입니다. 시연 시간에 에이전트를 학습시키는 것이 아니라, 학습된 에이전트를 곧바로 불러와서 시연해야 하므로 이러한 함수를 구현하도록 요구했습니다.
- **act(observation: Any, info: Dict):** 환경이 반환하는 값을 입력으로 삼아 행동을 출력력하는 함수입니다.

kymnasium.evaluate(env_id: str, agent: kymnasium.Agent)

구현한 강화 학습 에이전트를 시연하기 위한 클래스입니다. 다음과 같은 인자를 입력으로 받습니다:

- **env_id:** `kymnasium.make()` 등을 통해 환경을 불러올 때 사용하는 환경의 id를 의미합니다.
- **agent:** `kymnasium.Agent`를 상속하여 구현한 에이전트 구현체입니다.

Methods Only for Grid Crossing

물론, Grid Crossing 만을 위한 함수도 구현되어 있습니다. 바로, 에이전트 구현없이 여러분이 직접 플레이하는 것입니다.

kymnasium.grid_adventure.ManualPlayWrapper(env_id: str, debug: bool, **kwargs).play()

- **env_id:** `kymnasium.make()` 등을 통해 환경을 불러올 때 사용하는 환경의 id를 의미합니다.
- **debug:** True일 경우 각종 로그를 출력합니다

- **kwargs:** `gymnasium.make()`에서 `id` 이외에 필요한 인자들을 정의합니다 (예.
`render_mode = "human"`)

조작 방법은 다음과 같습니다:

- 방향키 $\leftarrow$: 플레이어가 좌측으로 90도 회전합니다.
- 방향키 $\rightarrow$: 플레이어가 우측으로 90도 회전합니다.
- 방향키 $\uparrow$: 플레이어가 보고 있는 방향으로 앞으로 1칸 이동합니다.

Environment

Environment Generation

Grid Crossing 환경은 일반적인 `gymnasium` 환경처럼 다음과 같이 생성할 수 있습니다:

```
import gymnasium as gym

env = gym.make(
    # Grid Crossing의 ID
    id='kymnasium/GridAdventure-Crossing-26x26-v0',
    # "human" 일 시 메인 스레드에서 게임 화면 렌더링,
    # "rgb_array" 일 시 백그라운드에서 게임 화면 렌더링
    render_mode='human',
    # "True" 일 시 BGM 재생; 오직 render_mode = "human"일 때만 동작
    bgm=True
)
```

Observation Space

미로 내의 객체들은 다음과 같습니다:

- : 일반적인 바닥입니다. 이곳은 마음대로 움직일 수 있어요.
- : 벽입니다. 이 곳으로는 이동할 수 없고, 플레이를 이리저리 헤매게 만듭니다.
- : 용암입니다. 이곳에 도착하면 게임 오버가 됩니다.
- : 목표 지점입니다. 이곳에 도착하면 미로를 탈출하게 됩니다.
- : 플레이어입니다. 화살표의 방향이 현재 향하고 있는 방향입니다.

에이전트는 이러한 미로의 구성 요소들을 [26, 26] Numpy 행렬로 알 수 있습니다. 행렬의 원소는 3자리 또는 4자리 숫자로 구성되어 있으며, 각 숫자는 다음을 의미합니다:

- 100: 바닥
- 250: 벽
- 900: 용암
- 810: 목표 지점
- 1000: 오른쪽으로 향한 플레이어
- 1001: 아래쪽으로 향한 플레이어
- 1002: 왼쪽으로 향한 플레이어
- 1003: 위쪽으로 향한 플레이어

Action Space

에이전트가 움직이도록 하기 위해서는 `Env.step()` 호출 시 다음 숫자 중 하나를 전달해야 합니다:

- 0: 왼쪽으로 90도 회전
- 1: 우측으로 90도 회전
- 2: 현재 보고있는 방향으로 1칸 전진

Reward Space

어떤 행동이든 보상의 양을 0으로 설정했습니다. 여러분들이 자유롭게 보상을 설계해보세요!

Episode End

목표 지점에 도달하거나, 용암에 빠지거나, 1,000번 이상 행동을 하게 되면 에피소드가 종료됩니다.

Submission

이루리에 제출해야 하는 파일은 두 개입니다.

Python for Training Your Own Agent (*.py)

아래 작성 예시를 복사하여, 자신만의 에이전트를 정의하고 이 에이전트를 훈련하는 코드를 구현한 후 제출하세요.

```
import gymnasium as gym
import kymnasium as kym
from typing import Any, Dict

'''
kymnasium.Agent를 상속하여
자신만의 에이전트를 구현
'''
class YourAgent(kym.Agent):
    def act(self, observation: Any, info: Dict):
        pass

    @classmethod
    def load(cls, path: str) -> 'kym.Agent':
        pass

    def save(self, path: str):
        pass

def train():
    '''
    Grid Crossing 환경은 다음과 같이 생성
    '''
    env = gym.make(
        id='kymnasium/GridAdventure-Crossing-26x26-v0',
        render_mode='human', # or "rgb_array"
        bgm=True # or False
    )
    '''
    여기서부터는 이 환경에 대해서 에이전트를 훈련시키는 코드를
    자유롭게 작성
    '''
```

Dependencies

에이전트 구현 시에 사용한 의존성 라이브러리의 목록을 제출해야 합니다.

가상 환경으로 **venv**를 사용하고 있다면, 다음 명령어를 실행하여 생성된 **requirements.txt** 파일로 제출해야 합니다.

```
pip list -format=freeze > requirements.txt
```

가상 환경으로 **conda**를 사용하고 있다면, 다음 명령어를 실행하여 생성된 **environment.yml**을 제출하세요.

```
conda env export > environment.yml
```

Due Date

- - 2025년 10월 13일 08:59

Evaluation

Code

RL Competition은 수업 시간에 다음의 코드를 실행하여 플레이한 결과로 결정됩니다 (별도의 실행 코드는 인정하지 않습니다):

```
import kymnasium as kym

'''
여러분이 정의하고 학습시킨 에이전트를 불러오는 코드를 넣으세요
예. agent = YourAgent.Load('some-path.pkl')
'''
agent = ...

kym.evaluate(
    env_id='kymnasium/GridAdventure-Crossing-26x26-v0',
    agent=agent,
    render_mode='human',
    bgm=True
```

)

Grading Criteria



Steps: 5 / Dist.: 7

최대한 적은 행동 횟수로 도착 지점에 도착하는 것이 목표이며, 세부적인 기준은 다음과 같습니다:

- 목표 지점에 도착하는 데 성공했을 시, **행동 횟수 (Steps)**가 적을수록 좋음
- 목표 지점에 도착하는 데 실패했을 시, **플레이어와 목표 지점간의 거리 (Dist.)**가 짧을수록 좋음

위 기준에 따라 순위를 매기며, 순위에 따른 점수는 다음과 같습니다:

- 1위 - 2위: 10%
- 3위 - 4위: 9%
- 5위 - 6위: 8%
- 7위 - 8위: 7%
- 8위 - 9위: 6%
- 10위 - 11위: 5%
- 12위 - 13위: 4%

0 Points Policy

단, 다음 중 하나에 해당하는 경우에는 무조건 0% 처리가 됩니다.

- 제출 마감 내에 제출물 (에이전트 구현체 및 requirements.txt)을 이 루리에 제출하지 않은 경우(지연 제출 또한 허용 안됨)
- requirements.txt에서 정의된 모든 의존성 라이브러리를 설치했음에도 불구하고 제출한 코드의 실행이 되지 않는 경우
- 제출한 코드의 실행 결과로 학습된 에이전트가 실제 시연 결과와 크게 다른 경우
- 시연을 하지 않거나 시연이 불가능한 경우 (자신의 팀이 시연을 하는 일자에 출석하지 않는 경우, 해당 학생만 0% 처리)
- 정책 또는 가치 함수를 학습하지 않고, 직접 임의로 지정한 정책 또는 가치 함수를 사용하는 경우

- 환경을 바꾸는 경우

Leaderboard

Index	Team Name	Clear	Steps	Dist.	Rank	Remark
1	123	O	85		12	
2	Ost	O	83		1	
3	Cherry	O	83		1	
4	Monte Carlo	O	83		1	
5	JSL게임개발원	O	83		1	
6	wqdsdsf	O	83		1	
7	봉구스밥버거	O	83		1	
8	우주선	O	83		1	
9	유재형	X	1000	2	13	
10	전형규	O	83		1	
11	조우석과 박경범	O	83		1	
12	카페모카	O	83		1	
13	김새연과 문경민	O	83		1	
특별	청강생들의 반란					