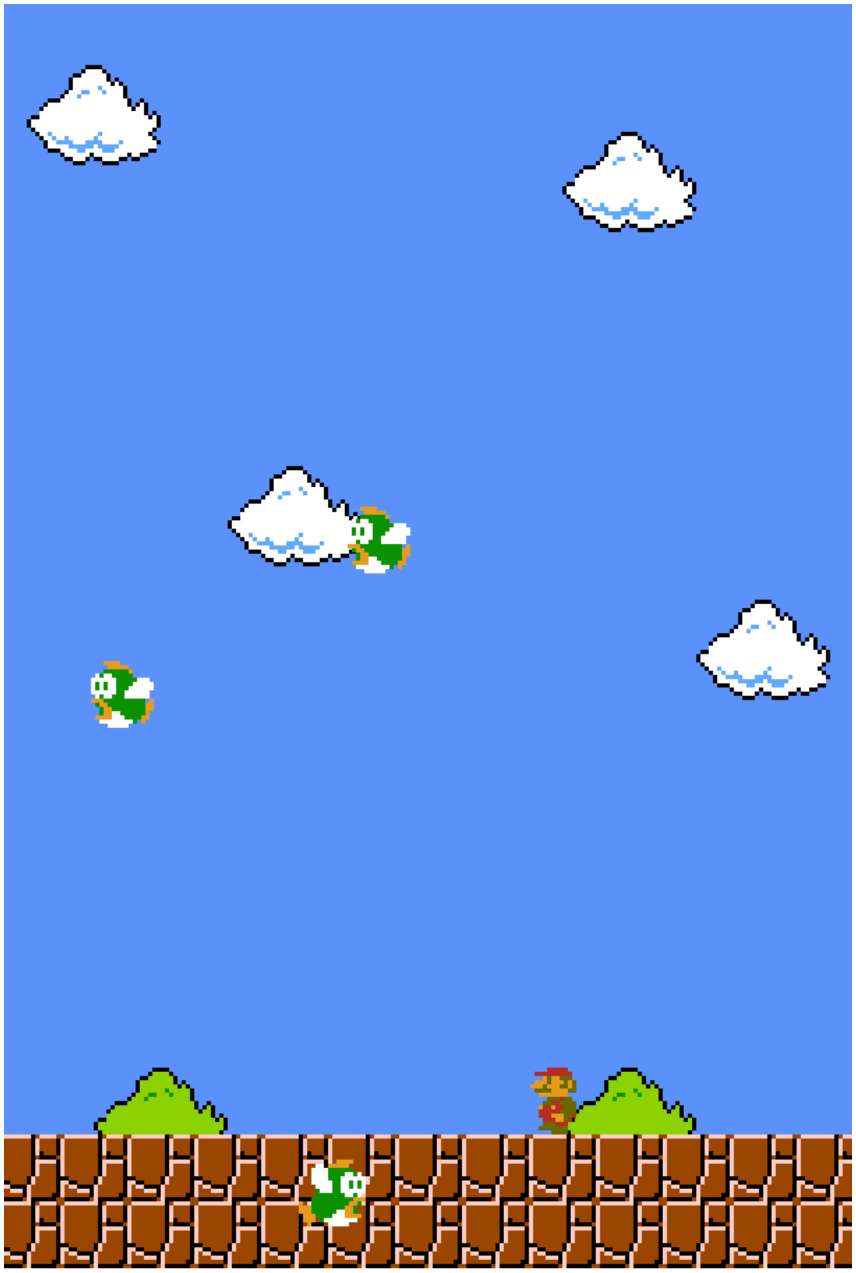
2026-02 KNU Advanced RL Competition Round 1

Avoid Blurp!



Time elapsed: 5.45 seconds

첫 번째 RL Competition은 하늘에서 떨어지는 보글보글Blurp을 피해 최대한 오랫동안 살아남는 **Avoid Blurp** 입니다!

Purpose

플레이어는 마리오를 조작해서 하늘에서 떨어지는 보글보글을 피해야 합니다. 단순해보이지만, 보글보글은 시간이 갈수록 더 많이 등장하고, 그 위치와 속도 또한 무작위로 변합니다. 여러분은 최대 3분 동안 하늘에서 떨어지는 보글보글의 폭격을 피하는 에이전트를 훈련해야 합니다.

Prerequisites

Python 3.12.x

Avoid Blurp은 Python 3.12 버전에서 동작하도록 구현되었습니다 (그 외의 버전에서는 동작을 보장할 수 없습니다)

Installation

Avoid Blurp과 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.

Brief Instruction for Gymnasium

Tutorial on Gymnasium & Dynamic Programming 에 강화학습 에이전트 훈련을 위한 gymnasium 환경에 대한 안내가 있지만, 확인하지 않은 학생들을 위하여 간단한 안내를 하겠습니다. Gymnasium은 강화 학습 에이전트 훈련을 위한 환경을 제공하는 Python 라이브러리로, Atari 게임, Multi-Joint Dynamics with Contact (MoJoCo; 로봇 제어 환경), Box2D (Pygame 기반 게임) 등의 다양한 강화 학습 훈련 환경을 제공하고 있습니다.

gymnasium.make

강화 학습 에이전트를 훈련시킬 환경인 gymnasium.Env 인스턴스를 생성하는 함수로, 생성할 환경의 ID와 환경별로 사용되는 인자를 선언하는 함수입니다. 예를 들어, 다음과 같이 환경을 생성할 수 있습니다:

```
import gymnasium as gym


env = gym.make(
    id="environment-id",
    render_mode="human", # or "rgb_array"
)
```

- id: 생성할 환경의 고유 구분자
- render_mode: 환경을 시각적으로 표시하는 방법을 설정. 환경마다 다르긴 하지만, 보통 다음과 같은 값을 가진다:
    - render_mode = "*human*": 게임을 메인 스레드에서 구동하며, 실제 게임 화면이 출력됨 (주의: Colab 및 Jupyter Notebook에서는 실행 안 됨)
    - render_mode = "*rgb_array*": 게임 화면을 백그라운드에서 렌더링; 실제 게임 화면은 나오지 않으나, 게임 진행은 되고 있음

보통, 강화학습 에이전트를 훈련시킬때는 render_mode를 "*rgb_array*"로 설정하시면 됩니다. 실제 게임 화면을 메인 스레드에서 렌더링하지 않으므로 플레이하는 시간이 더 빠르기 때문입니다. 반대로, *render_mode*를 "*human*"으로 설정하는 것은 시연이나 직접 환경을 눈으로 확인하고 싶을 때입니다.

gymnasium.Env

Gymnasium의 핵심인 환경의 구현체로, 모든 환경 구현체가 다음과 같은 핵심 함수를 가지고 있습니다:

- Env.step(action): 행동인 action을 입력으로 받아서 실제로 환경에 행동을 수행한 후 다음 상태와 보상을 반환하는 함수로, 아래와 같은 결과를 튜플로 반환:
    - observation: 행동으로 인해 전이된 새로운 상태

    - `reward`: 행동에 대한 보상
    - `terminated`: 종료 상태Terminal State 도달 여부를 반환
    - `truncated`: 종료 상태 도달 이외의 종료 상황 여부를 반환
    - `info`: 그 외 환경에 대한 정보
- `Env.reset(seed)`: (무작위성이 존재하는 환경이라면 Random Seed인 `seed`를 입력으로 받아) 환경을 초기 상태로 되돌리는 함수로, 아래와 같은 결과를 튜플로 반환
    - `observation`: 초기 상태
    - `info`: 그 외 환경에 대한 정보
- `Env.render()`: 환경의 상태를 (보통은) Numpy 배열 이미지로 반환
- `Env.close()`: 현재 환경을 종료

위와 같은 핵심 함수 외에도, 환경에 대한 기본적인 정보를 알려주는 속성 또한 존재합니다.

- `Env.action_space`: 환경에서 가능한 행동 공간Action Space을 반환
- `Env.observation_space`: 환경의 상태 공간State Space을 반환

Brief Instruction for Kymnasium

kymnasium 환경은 KNU Gymnasium의 준말로, 강원대학교 강화학습 수업에서 활용할 커스텀 Gymnasium 환경입니다. 구현에 신경 써야 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, act입니다.

```
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, `Python`의 기본 직렬화 라이브러리인 `pickle` 등을 활용 가능합니다.
- **`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 출력하는 함수입니다.

`kymnasium.evaluate(env_id: str, agent: kymnasium.Agent, bgm: bool, seed: int)`

구현한 강화학습 에이전트를 시연하기 위한 함수입니다. 다음과 같은 인자를 입력으로 받습니다:

- **`env_id`**: `gymnasium.make()` 등을 통해 환경을 불러올 때 사용하는 환경의 id를 의미합니다.
- **`agent`**: `kymnasium.Agent`를 상속하여 구현한 에이전트 구현체입니다.
- **`bgm`**: BGM 재생 여부를 결정합니다.
- **`seed`**: 환경의 Random Seed를 결정합니다. 시연 당일날 정해지며, 모든 학생이 동일한 Random Seed를 활용해서 시연을 합니다.

Environment

그럼, 이번 과제에서 사용할 Avoid Blurp 환경에 대해 설명하겠습니다.

Generation

Avoid Blurp 환경은 일반적인 gymnasium 환경처럼 다음과 같이 생성할 수 있습니다:

```
import gymnasium as gym


env = gym.make(
   # Avoid Blurp의 ID
   id='kymnasium/AvoidBlurp-Discrete-Ballistic-Normal-Stage-1',
   # "human": 메인 스레드에서 게임 화면 렌더링,
   # "rgb_array": 백그라운드에서 게임 화면 렌더링
   # "none": 게임 화면 렌더링 없이 내부 로직만 모델링; 에이전트 훈련에
             활용
   render_mode='human',
   # "True" 일 시 BGM 재생; 오직 render_mode = “human”일 때만 동작
   bgm=True
)
```

Observation Space

에이전트는 마리오와 보글보글의 속성들을 각각 *mario*와 *blurps*라는 키를 갖고 Numpy 행렬을 값으로 가지는 딕셔너리로 알 수가 있습니다.

```
{
   "mario": [좌측 좌표, 위쪽 좌표, 우측 좌표, 아래쪽 좌표, X축 속도, Y축
속도, 점프 여부],
   "blurps":  [
      [좌측 좌표, 위쪽 좌표, 우측 좌표, 아래쪽 좌표, X축 속도, Y축 속도,
Y축 가속도],
      [좌측 좌표, 위쪽 좌표, 우측 좌표, 아래쪽 좌표, X축 속도, Y축 속도,
Y축 가속도],
      ...
   ]
}
```

*mario*는 플레이어가 조작하는 마리오의 속성을 7개의 원소를 가지는 Numpy 행렬로 표시합니다.

*blurps*의 속성은 [30, 7] Numpy 행렬로 제공되며, 각 행마다 화면에 표시된 보글보글의 속성을 7개의 원소로 표현합니다. 화면에 표시되지 않은 보글보글의 경우 모든 원소의 값이 0이며, 좌표는 화면 좌측 최상단이 각각 0, 0입니다.

Action Space

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

- 0: 움직이지 않습니다.
- 1: 마리오가 좌측으로 조금 이동합니다.
- 2: 마리오가 우측으로 조금 이동합니다.
- 3: 마리오가 제자리에서 점프합니다.
- 4: 마리오가 왼쪽으로 점프합니다.
- 5: 마리오가 우측으로 점프합니다.

Reward Space

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

Info

현재까지 경과한 시간과 게임 상태를 다음과 같이 반환합니다:

```
{
   "time_elapsed": XXXX,
   "game_state": XXXX
   # "init": 초기 상태
   # "playing": 일반적인 플레이 중
   # "game_over": Blurp에 닿아서 게임 종료
   # "cleared": 주어진 시간 동안 살아남음
}
```

Episode End

보글보글과 마리오가 충돌하거나, 3분 동안 보글보글을 피한 경우 에피소드가 종료되며 반환되는 값은 다음과 같습니다:

- 보글보글과 마리오가 충돌한 경우:
    - `terminated = True`, `truncated = False`
    - `info['game_state'] = 'game_over'`
- 2분 동안 보글보글을 피한 경우:
    - `terminated = True`, `truncated = False`
    - `info['game_state'] = 'cleared'`

Manual Play

`kymnasium.avoid_blurp.ManualPlayWrapper(env_id: str, debug: bool, **kwargs).play(seed)`를 활용해 직접 플레이해 볼 수도 있습니다:

- **`env_id`**: *`gymnasium.make()`* 등을 통해 환경을 불러올 때 사용하는 환경의 `id`를 의미합니다.
- **`debug`**: *`True`*일 경우 각종 로그를 출력합니다
- **`seed`**: Random Seed를 설정합니다.

조작 방법은 다음과 같습니다:

- 방향키 ←: 마리오가 좌측으로 조금 이동합니다.
- 방향키 →: 마리오가 우측으로 조금 이동합니다.
- 스페이스바: 마리오가 점프합니다.

Evaluation

Competition (20%)

Code

RL Competition은 수업 시간 중 본인의 기기 (노트북, 원격 연결 등)에서 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/AvoidBlurp-Discrete-Ballistic-Normal-Stage-1',
    agent=agent,
    render_mode='human',
    bgm=True,
    seed=... # 시연 당일 설정
)
```

Grading Criteria



보글보글을 피해 최대한 오래 버티는 게 목표이며, **총 2번 시도 중 가장 오래 버틴 기록** 순으로 순위를 매깁니다. 순위에 따른 점수는 다음과 같습니다:

- 1위: 20%
- 2위: 19%

- 3위: 18%
- 4위: 17%
- 5위: 16%
- 6위: 15%
- 7위: 14%
- 8위: 13%
- 9위: 12%
- 10위: 11%
- 11위: 10%
- 12위: 9%
- 13위: 8%
- 14위: 7%
- 15위: 6%
- 16위: 5%

Presentation (10%)

Purpose

Competition 의 결과 뿐만 아니라, 여러분이 환경을 어떻게 분석했고, 어떤 강화학습 방법을 선택했으며, 무엇이 성공하거나 실패했는지를 공유하여 다른 학생들이 더 좋은 에이전트를 구현할 수 있도록 돕는 5분 내외의 발표도 하셔야 합니다.

Grading Criteria

- 환경 및 문제 분석 (2%): 주어진 환경을 해결하기 위한 상태, 행동, 보상 등의 처리 방법
- 강화학습 방법의 이해와 선택 근거 (3%): 알고리즘, 모델 구조, 탐험 방법, 주요 하이퍼파라미터의 선택 근거와 대안과의 차이
- 다른 학생에게 제공하는 실질적 도움 (3%): 다른 학생이 적용할 수 있는 구체적인 개선 방법과 적용 조건, 예상 효과, 구현 시 주의점, 재현을 위한 설정 및 학습량 등

- 한계 및 개선점 (1%): 현재 적용한 방법의 한계와 그 원인을 분석하고, 개선을 위한 아이디어 제공
- 전달력과 시간 활용 (1%): 5분 안에 핵심을 논리적이고 명확하게 전달

Peer Evaluation Only

발표는 교수가 평가하는 대신 수업을 듣는 다른 학생들이 평가합니다. 발표의 목적은 다른 학생들이 ***"이 발표를 듣고 나서 내가 구현하거나 실험해 볼 수 있는 구체적인 개선 방법이 생겼는가?"***를 느꼈는지 평가하기 위함이기 때문입니다. 따라서, ***"ChatGPT가 그랬는데요"*** 등의 내용은 학생들에게 도움이 되지 않으므로 허용되지 않습니다.

Submission

이루리에 제출해야 하는 파일은 총 세 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():
    '''
    Avoid Blurp 환경은 다음과 같이 생성
    '''
    env = gym.make(
        id='kymnasium/AvoidBlurp-Discrete-Ballistic-Normal-Stage-1',
        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 venv를 사용하고 있다면, 다음 명령어를 실행하여 생성된 requirements.txt 파일로 제출해야 합니다.

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

가상 환경으로 conda를 사용하고 있다면, 다음 명령어를 실행하여 생성된 environment.yml을 제출하세요.

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

Presentation

수업 시간에 발표할 발표자료를 PDF 파일로 업로드하세요.

Due Date

- - 2026년 10월 28일 08:59

0 Points Policy

단, 다음 중 하나에 해당하는 경우에는 무조건 0% 처리가 됩니다.

- 제출 마감 내에 제출물 (에이전트 구현체, `requirements.txt`, 발표자료)을 이루리에 제출하지 않은 경우(지연 제출 또한 허용 안됨)
- `requirements.txt`에서 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
- 정책 또는 가치 함수를 학습하지 않고, 임의로 만든 (Hard-Coded) 정책 또는 가치 함수를 사용하는 경우
- 환경을 바꾸는 경우