University of Western Australia

CITS3001

Agents, Algorithms and Artificial Intelligence

Super Mario Project

September 12, 2023

Figure 1: Sourced from XKCD [1]

Contents

1 Project Overview

In this project, you will develop AI agents to control the iconic character Mario in the classic game Super Mario Bros using the gym-super-mario-bros environment. The main objective is to implement at least two distinct AI algorithms/methods and compare their performance, strengths, and weaknesses in the context of playing the game.

You can undertake this project in teams of 2 that you select, if you are looking for a partner please reach out to your lab demonstrators by emails

1.1 Requirements

1.1.1 Gym-Super-Mario-Bros Environment Setup

- Set up the gym-super-mario-bros [2] environment on your local machine or any designated platform, see 4.2.1Environment Creation.

Figure 2: Example gym-super-mario-bros

As gym-super-mario-bros requires a graphics environment it will not be possible to run it through WSL, so a Windows native python installation will be required. Macos and Linux users should have no issues.

1.1.2 AI Algorithm Implementations

- Choose and experiment with at least two AI algorithms. You may wish to consider the following:
  - Reinforcement Learning: Q-learning [3], TD(λ) [4]
  - Rule-Based AI: logic and heuristics.
  - Monte Carlo Tree Search (MCTS) [5]

You are welcome to use more advanced algorithms that utilise deep learning such as DQN's [6] or Proximal Policy optimisation etc. but these are not covered in the unit and lab facilitators may not be able to assist with your implementations. These algorithms will also have to be referenced in your project report.

You are allowed to use existing implementations such as those from stable baselines however **you are required to implement at least one of the algorithms yourself**. For example the following combinations of algorithms would be allowed (this list is not exhaustive):

- Hand Implemented Rule based agent and PPO from Stable Baselines
- Hand Implemented TD(λ) and Hand Implemented DQN

but comparing PPD and DQN from stable baselines would not be allowed.

You are welcome to used utilities and libraries from Stable Baselines in your own implementations, just make note of them in your report. An algorithm from stable baselines can be counted as hand implemented if sufficient fine tuning, adjustments or optimisations have been made for the Super Mario Bro's environment, but you will have to make note of these in your report and you may be required to explain them in an interview see 2.

For example if you were to take the DQN from Stable baselines, define your own custom policy, have custom image prepossessing and added internal replay it could count as hand implemented.

Projects that use two hand implemented algorithms are likely to score higher on the implementation section of the marking criteria.

1.2 Final Project Report

To demonstrate your understanding of your implementations you will be required to write a final project report. Your report must conform to the following guidelines

- At least 3 pages
- No longer than 5 pages
- No smaller than size 12 font

You are allowed to add appendices with extra figures and words but these may not be marked. Your report should cover the following areas:

1.2.1 Analysis

- Analyze and contrast the performance of the chosen AI methods.
- Discuss their respective strengths, weaknesses, and suitability for playing Super Mario Bros.

To analyze and contrast your chosen AI methods you should design experiments to highlight their differences. The simplest and most obvious experiment will be the agents **Performance on Predefined levels** i.e. the levels the agents were trained on. You can then design more specific experiments for your choice of algorithms. For example, consider two popular Machine learning approaches Reinforcement Learning (RL) and Genetic Algorithms (GAs). RL, exemplified by algorithms like Deep Q-Networks (DQN), learns from trial and error, using a reward system to guide decision-making. This method excels in complex environments with high-dimensional state spaces, making it well-suited for Super Mario Bros. Its strength lies in its adaptability to diverse tasks and ability to handle dynamic environments. A few possible experiments that highlight this difference:

- **Generalisation to Unseen Levels**: Take the RL and GA agent and compare how far right they get on 5 different levels. If the RL agent makes it further this is evidence that it is better at generalising over different environments
- **Transfer Learning**: train both agents simplified version of the game (e.g. the v3 gym) and then fine-tune it on the full game (the v0 gym) and compare overall performance.

However, RL can be computationally expensive and may require extensive training time. On the other hand, Genetic Algorithms, a population-based optimization technique, evolve a population of candidate solutions over generations can also be expensive to train. Some experiments to compare the overall efficiency of the algorithms could be:

- **First to complete the level**: Compare how many training iterations it takes for each agent to complete the first level. Then how many additional iterations it takes to beat level 2. This could be plotted as a level progress vs training time curve.
- **Model scaling**: Compare how the algorithms performance responds to changes in parameters or hardware. Does changing the GA population size have an effect? Does increasing the training steps for RL make a difference?

The above experiments are intended to be examples, you are not required to cover these exact experiments in your report. However, you are welcome to use them as inspiration for your own comparisons. It is strongly recommended you design at least one experiment that highlights a disparity between your two chosen algorithms.

1.2.2 Performance Metrics

You will notice that gym-super-mario-bros reward function assumes the objective of the game is to move as far right as possible. You are encouraged to come up with other performance and evaluation metrics for your agents. Novel and interesting metrics that you come up with will be rewarded. Some examples of other performance metrics that you may want to consider in your experiments are:

- How many different levels an algorithm complete without additional training
- How many training iterations it takes for an agent to complete a level

- The number of actions the algorithm takes to complete a level
- How many deaths did an agent encounter during training
- How many coins an agent collects

You can then uses these metrics in your analysis 1.2.1 and the experiments you design to contrast your two algorithms.

1.2.3 Visualization and Debugging

- Include what visualization techniques you used to gain insights into the agent's decision-making process.
- Include what debugging/profiling tools you utilised to optimize the algorithms and enhance performance.

The following are examples of visualisation and debugging could be used for the example experiments in the Analysis 1.2.1 section:

1. **Performance on predefined levels**
    - Visualization: Utilize heatmaps to visualize the areas of the level where the agent spends the most time, indicating which parts are particularly challenging. Additionally, use line plots to track the agent's progress over time.
    - Debugging/Profiling Tools: Employ a Python profiler (e.g., cProfile) to identify bottlenecks in the RL training process, ensuring that computational resources are used efficiently.
2. **Generalisation to Unseen Levels**
    - Visualization: Create side-by-side comparisons of the agent's performance on seen vs. unseen levels. Have a screen capture of how the two different agents responded.
    - Debugging/Profiling Tools: Monitor memory usage and execution times using tools like memory profilers (e.g., memory profiler) and time profilers (e.g., timeit).
3. **Transfer Learning**
    - Visualization: Present performance metrics (completion time, score) for both the simplified and full game setups. Use visual aids like line charts to show the agent's improvement over training iterations.
    - Debugging/Profiling Tools: Monitor memory usage and computational resources during both the initial training and fine-tuning stages to ensure efficient transfer learning.
4. **Model Scaling**
    - Visualization: Create bar charts comparing the training times. Include information on hardware specifications to provide context for computational resource usage.
    - Debugging/Profiling Tools: Employ system monitoring tools (e.g., htop) to track CPU/GPU usage and memory allocation during training.

1.3 Marks Distribution

- AI Algorithm Implementations (40%)
  - Successful implementation of two or more AI methods (20%)
  - Code quality, readability, and efficiency (10%)
  - Integration with the gym-super-mario-bros environment (10%)
- Report Comparison and Contrast (30%)
  - In-depth analysis of the algorithms' strengths and weaknesses (15%)
  - Properly conducted experiments and results presentation (10%)
  - Effective comparison of AI methods (5%)
- Performance Metrics and Visualization (20%)
  - Selection and definition of appropriate performance metrics (10%)
  - Quality of visualization techniques (10%)
- Report Quality (10%)
  - Clarity and organization of the report (5%)
  - Overall presentation and writing quality (5%)

1.4 Additional Notes

- Regularly check in with the instructor for progress assessments and guidance during the project timeline.
- Ensure that your work is original and properly referenced to maintain academic integrity throughout the project.

1.5 Clarifications

- any python packages or libraries are permitted, just make sure they are included in your environment file when you submit
- You are permitted to use other gym-super-mario-bros environments like **SuperMarioBros-v3** see 3. However, implementations that solve the original v0 gym will be looked upon favourably.

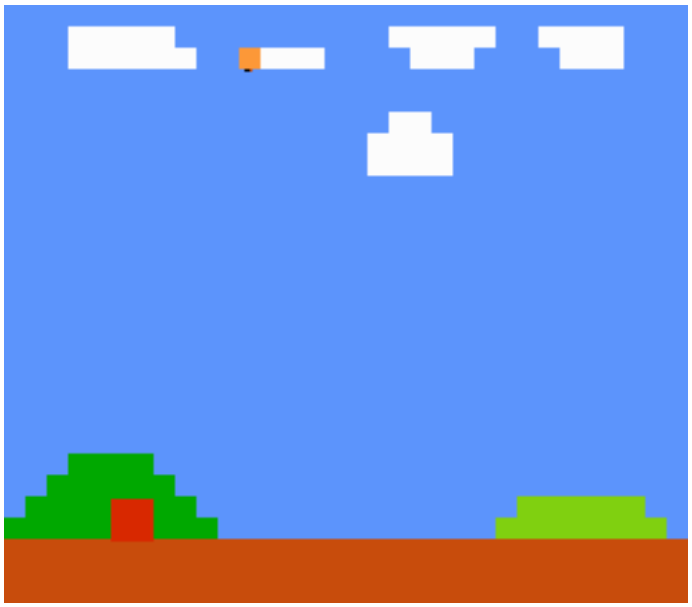

Figure 3: SuperMarioBros-v3 example environment which is allowed

- The implementations 1.1.2 section has been updated. You are now allowed have one of your algorithms be from Stable Baselines and one of them be hand implemented. Please make note of the specifics
- The Project report section 1.2 has been updated to include more examples of metrics and comparisons

2 Submission Guidelines

You will be required to submit the following:

- A zip file containing the code for your agents and any model files they might use
- Your final report as a PDF file (please no word docs)

Please ensure your code submission contains a README.md file explaining how to setup the environment for your project and how to run the two separate agents individually Failure to have properly running agents will result in an interview.

3 Interviews

If markers suspect instances of plagarism or excessive reliance on ChatGPT or any external sources, a procedural measure has been established. Students found in such situations may be requested to attend an interview with their respective lab instructor. The purpose of this interview is to provide students with an opportunity to demonstrate their understanding of the project. It is worth noting that consistent attendance in laboratory sessions significantly mitigate the likelihood of being selected for an interview.

4 Environment Setup

You are using an official public package, so please don't spam the github issues. I have created a fork for our unit on github that you can file specific requests under

4.1 Conda

Anaconda Python is the recommended method of installation for this project

4.1.1 Linux

Download installer from anaconda
you may need to run

```
conda init bash
```

depending on your shell

4.1.2 Macos

follow the instructions on anaconda official documentation

4.2 Windows

firstly follow the instructions on anaconda official documentation

You will need to install the gym with pip later, which will require the microsoft build tools.

So install the following

- visual studio (not to be confused with VS-Code)
- visual studio build tools

the build tools are mandatory and need to be installed before installing the gym.

4.2.1 Environment Creation

Open anaconda prompt on windows (type anaconda in the search bar), don't open powershell or command prompt as it may have difficulty finding the conda program.

Then create the new conda environment with:

```
conda create -n mario python=3.8
```

then activate the environment.

```
conda activate mario
```

4.3 Package Installation

There isn't a working conda package for gym-super-mario-bros so it is best to install with pip, this package will be installed inside the conda environment if you have it activated

```
pip install gym-super-mario-bros
```

4.4 Poetry

You only need to use one package manager, so use conda or poetry not both
Poetry has been less successful so use at your own risk per platform.

4.4.1 Tool Installation

See poetry documentation on how to install for your system.

```
curl -sSL https://install.python-poetry.org | python3 -
```

4.4.2 Package Installation

```
poetry add gym-super-mario-bros
```

```
poetry install
```

5 Test Code

```
from nes_py.wrappers import JoypadSpace
import gym_super_mario_bros
from gym_super_mario_bros.actions import SIMPLE_MOVEMENT
import gym

env = gym.make('SuperMarioBros-v0', apply_api_compatibility=True, render_mode="human")
env = JoypadSpace(env, SIMPLE_MOVEMENT)

done = True
env.reset()
for step in range(5000):
    action = env.action_space.sample()
    obs, reward, terminated, truncated, info = env.step(action)
    done = terminated or truncated

    if done:
       state = env.reset()

env.close()
```

6 Running Test Code

6.1 Windows

Due to the graphics library requirements of gym-super-mario-bros it will be challenging to use WSL to complete this project. As such a native windows approach is recommended.

6.2 VS-Code (reccomended)

Use **Ctrl-Shift-p** to open up the command prompt and run

```
Terminal: Select default profile
```

and change it to command prompt. Anaconda prompt will not be available as an option, so don't worry. We are changing this as the default option for windows is powershell which doesn't play nicely with conda environments.

Next open up the command prompt again and run

```
Python: Select Interpreter
```

and change this to the conda or poetry environment you have created (this assumes you have already created one, if not see the section on environment creation)

With this configured you should be able to click the run button in the top left hand corner on any test code. If you get and import error try pressing the run button again as there have been issues where the python file is run before the conda environment has been activated.

6.3 Command Prompt

Use anaconda prompt (not powershell or command prompt) to run the following

```
conda activate mario
```

Then run the test code (assuming your file is called test.py)

```
python test.py
```

6.4 Linux/Macos

On a terminal make sure you have the right conda environment activated

```
conda activate mario
```

Then run the test code

```
python test.py
```

7 Example Issues and How to Fix Them

7.1 Tuple unpacking API Issue

```
DeprecationWarning: ‘np.bool8‘ is a deprecated alias for ‘np.bool_‘.  (Deprecated NumPy 1.24)
Traceback (most recent call last):
  File "/home/dadams/Desktop/Other/super-mario/test.py", line 11, in <module>
    state, reward, done, info = env.step(env.action_space.sample())
                                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "site-packages/nes_py/wrappers/joypad_space.py", line 74, in step
    return self.env.step(self._action_map[action])
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "gym/wrappers/time_limit.py", line 50, in step
    observation, reward, terminated, truncated, info = self.env.step(action)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: not enough values to unpack (expected 5, got 4)
```

This issue was caused by an api cmpatability error so simply put:

```
env = gym.make('SuperMarioBros-v0', apply_api_compatibility=True, render_mode="human")
```

References

[1] Randall Munroe. Seashell. `https://xkcd.com/1236/`, 2013. [Online; accessed October 20, 2019].

[2] Christian Kauten. Super Mario Bros for OpenAI Gym. GitHub, 2018.

[3] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[4] Harm Seijen and Rich Sutton. True online td (lambda). In *International Conference on Machine Learning*, pages 692–700. PMLR, 2014.

[5] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.