ARI 510 Lab 4
Ryan Smith
12/3/24

# Lab 4

Cartpole and Lunar Landing Reinforcement Project.

# Challenges Faced During Training

The overall training process for a reinforcement learning project was essentially new to me, so that in itself was somewhat of a challenge at the start. I hadn't used the Gymnasium library, so that was somewhat of a challenge also at the start. There wasn't a ton of code involved, so that didn't seem to be much of a challenging component. The example cartpole code explained things well, and the process was fairly similar for Lunar Lander, which is the other system I chose.

Computational Resources wise: I was able to run the training on my own system, which is not the best system. A Mac mini M4, which is new, but it is a base model, not the pro model. The pro models have a different GPU architecture which is allegedly much better for AI related programs. For example, the pro model is considerably faster with Ollama. I have tested Ollama on an older M2 MacBook Pro, and it is faster than my new standard base model M4 Mac mini. I setup the Great Lakes Cluster account, but didn't seem to need it for this lab. My system was running the code fast enough. Many of the training phases took an hour or two at a time, which was acceptable.

I was generally using 10,000 episodes. The time for these ranged from 45 minutes to 3 hours. Batch size and update_target_every seemed to be the main hyperparameters affecting the computation time. A related concept here is diminishing return. Numbers in general from the tests stabilized so the diminishing return was high, not really warranting an increased number of episodes and need for more computational resources, was my opinion. They either stabilized or fluctuated consistently. I imagine a more complex project other than lunar lander might have benefitted more from more computation resources.

# Overall Lessons Learned

I had almost no experience with reinforcement learning before this lab. This lab taught me a ton about the topic. It also helped me grasp the overall concepts of machine learning, with finally working with a reinforcement learning project and being able to compare it to supervised learning and unsupervised learning.

I hadn't used OpenAI's Gymnasium library also, which is what Cartpole and Lunar Lander are from. Working with them was a great learning experience in general, and both of those projects seem very well put together and help explain the concepts of reinforcement learning well. I haven't checked out the others, but I will in time. I noticed another thing with this lab: I seemed to learn the concepts somewhat quicker than before, which I attributed to an increased overall understanding of AI, and mainly supervised learning, and to a lesser extent unsupervised learning. The general foundation knowledge helped me grasp the concepts quicker.

# Hyperparameters Adjusted and Their Impact

gamma = 0.99
batch_size = 64
epsilon = 1.0
epsilon_min = 0.1
epsilon_decay = 0.995
update_target_every = 100

The ones listed above were my initial training settings.

Lunar Lander benefited from a high emphasis on long-term rewards, a more gradual exploration decay, and a stable target network.

Here is some initial observations:

- The success rate starts improving significantly after around 300 episodes.
- The average reward fluctuates and shows improvement over time but still has negative values in many episodes.
- The number of steps per episode increases, indicating that the agent is learning to survive longer but may not be optimizing for successful landings consistently.

Here is a screenshot of the end of the training phase, which took about an hour, with 10,000 episodes.

```
Episode 9950, Reward: 282.200522881578, Avg Reward (last 50): 141.01575979141788, Steps: 185, Avg Steps: 269.96, Success Rate: 112.04
Training complete.
```

As a note, my initial code was not percentage for success rate, that is why at the end is is showing 112. I fixed this for later tests, and will go over that later in this section.

Some changes made:

```
gamma = 0.99
batch_size = 128  # Increased batch size
epsilon = 1.0
epsilon_min = 0.1
epsilon_decay = 0.990  # Faster decay rate
update_target_every = 50  # More frequent updates
```

The main changes here are of batch_size to 128 and epsilon_decay and update_target_every. The model might have been exploring too much, and the decay change will increase exploitation. Batch size gives move experience in each training step, potentially stabilizing things. And the update_target_every helps stabilize things also. Those were the reasoning of the change. The results are below:

Worse results. Success rate was lower, rewards were lower, and the computation time increased tremendously.

Another set of changes:

Gamma to .95
Epsilon min to .05
Epsilon to .9

The gamma change puts more emphasis on immediate rewards. This should help the agent learn faster.

Lower epsilon lowers the initial exploration phase, increasing the exploitation.

Lower epsilon min is maintaining some exploration though.

Results:

The changes performed very poorly. The success rate just fluctuated at about 2%, along with the other numbers. It never stabilized. The rewards also remained negative, and fluctuated.

Next configuration for hyper parameters:

```
# Hyperparameters
gamma = 0.99
batch_size = 32
epsilon = 1.0
epsilon_min = 0.1
```

epsilon_decay = 0.995
update_target_every = 100
rolling_window = 250  # Updated to sync with video generation

This performed well.  The batch_size might be a major factor. Lowering this to 32 is increasing the computation speed, and the success rate considerably.  The first 2,500 episodes fluctuated considerably, then began a steady increase in success rate, along with average rewards.

Here is a screenshot of the turning point of 10,000 episodes.

```
eps: 269, Avg Steps: 447.16, Success Rate (last 250): 2.80%, Overall Success Rate: 3.77%
ure video compatibility with most codecs and players. To prevent resizing, make your input
eps: 167, Avg Steps: 449.904, Success Rate (last 250): 4.40%, Overall Success Rate: 3.85%
ure video compatibility with most codecs and players. To prevent resizing, make your input
: 1000, Avg Steps: 544.236, Success Rate (last 250): 7.60%, Overall Success Rate: 4.26%
ure video compatibility with most codecs and players. To prevent resizing, make your input
s: 1000, Avg Steps: 623.712, Success Rate (last 250): 14.80%, Overall Success Rate: 5.32%
ure video compatibility with most codecs and players. To prevent resizing, make your input
eps: 621, Avg Steps: 657.84, Success Rate (last 250): 38.00%, Overall Success Rate: 8.29%
ure video compatibility with most codecs and players. To prevent resizing, make your input
s: 1000, Avg Steps: 665.624, Success Rate (last 250): 49.60%, Overall Success Rate: 11.73%
ure video compatibility with most codecs and players. To prevent resizing, make your input
os: 591, Avg Steps: 554.424, Success Rate (last 250): 46.40%, Overall Success Rate: 14.40%
ure video compatibility with most codecs and players. To prevent resizing, make your input
```

The midpoint there is at about 2,500 episodes of 10,000.  The ending accuracy was 53%.  The average rewards was up to 63, but this appeared to be stabilizing.  It is possible that diminishing returns hadn't really become a factor yet for accuracy though, and the 53% could have increased more, but the increase appeared to be slowing down..

I tested some other settings of lower batch size and high update_target_every but it didn't improve the scores.  Other various configurations were tested also, but the best one was the one listed prior to this, with the screenshot.

# A Comparison of Cartpole and Lunar Lander

CartPole and Lunar Lander are both classic reinforcement learning environments that serve as benchmarks for testing AI agents, but they differ significantly in their complexity. CartPole, with its simple objective of balancing a pole on a moving cart, is like a beginner's puzzle. It has a smaller state space, only requiring the agent to track the cart's position and velocity, and the pole's angle and velocity. The agent makes simple decisions, choosing only to push the cart left or right. It is also considered a continuing task, unless there is a crash, which would make it like

an episodic task.  Lunar Lander is an episodic task.  Episodic tasks have a clear ending.  One of the main differences here is that Lunar Lander has fuel, where Cartpole doesn't, so it can go indefinitely.

Lunar Lander is more like a challenging flight simulator. The goal is to land a spacecraft safely, requiring the agent to control its horizontal and vertical movement, angle, and velocity, all while conserving fuel.  This translates to a larger state space and a more complex action space, with options to fire different engines or do nothing. The dynamics involved, such as gravity and inertia, make it a significantly harder problem to solve, demanding more sophisticated algorithms and strategies from the learning agent. While CartPole focuses on basic balance and avoiding overcorrection, Lunar Lander introduces the complexities of coordinated movement and resource management, making it a more demanding testbed for advanced reinforcement learning agents.