Machine Learning Engineer Nanodegree Capstone Proposal

Scott Lilleboe
December 18, 2018

Proposal

Domain Background

The capstone proposal involves the creation of an agent that will land the OpenAI gym Lunar Lander under established metrics that will determine success. The Lunar Lander simulation utilizes the OpenAI gym library which is a collection of environments to test reinforcement learning algorithms (OpenAI, Getting Started with Gym, 2018).

I have an interest with various NASA projects, especially with planetary exploration. This particular environment relates to recent Mars exploration such as the Insight mission (NASA, 2018).

Problem Statement

The Lunar Lander problem comprises an 8-dimensional continuous state space and an action space of two real values vectors from -1 to +1. The landing pad is static at coordinates (0, 0) and the coordinates are the first two numbers in state vector. The reward for moving from the top of the screen to landing pad with zero speed is between 100 to 140 points. The episode finishes if the lander crashes or comes to rest, receiving an additional +-100 points. Each leg to ground contact is worth an additional 10 points. Firing the main engine is worth -0.3 points each frame. The fuel for the lander is infinite. (OpenAI, LunarLanderContinuous-v2, 2018)

Datasets and Inputs

The inputs to the algorithm are provided by the given Lunar Lander code. The input values of this state space are a deterministic 8-dimensional continuous state space.

Solution Statement

A successfully defined solution is when a model during training is returned an average score of 200 or more for 5 consecutive training episodes.

Benchmark Model

The chosen bench mark model will be a single 128 layer deep Q-learning network (DQN). The deep neural network (DNN) portion of the algorithm will compile with a MSE loss parameter utilizing the Adam optimizer with the default learning rate.

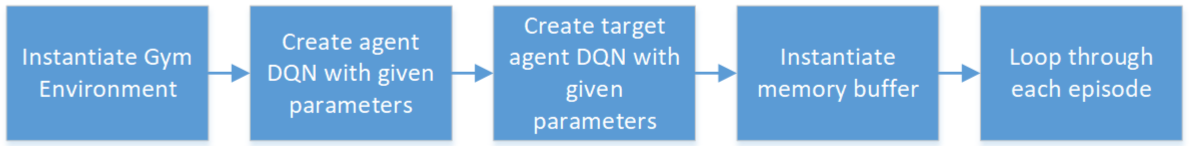
Evaluation Metrics

The benchmark and solution model will be evaluated over two metrics. The first metric will be the total average reward of a trained model over 100 iterations. This metric will give credit to the solution that had a better solution in terms defined by the Problem Statement. The second metric is the number of iterations needed to successfully train the model and reach a solution as defined in the Solution Statement section.

The reason for the second metric is to give recognition to a solution that was successful and may have a substantially quicker time to train.

Project Design

The project will utilize a deep Q-learning network (DQN) with a replay buffer. The flow of the project will be as follows:


Instantiate Gym Environment
Create agent DQN with given parameters
Create target agent DQN with given parameters
Instantiate memory buffer
Loop through each episode

Below is pseudocode of the last step from above, the looping of each episode:

```
agent = DQN
target_agent = DQN
decay = 0.0001
epsilon = 0.99

for episode in episodes:
    state = environment.reset()
    total_reward = 0
    for step in steps:
        action = random_integer(0, length(actions))
        next_state, reward, done = environment.step(action)
        total_reward += reward
        replay_memory.add((state, action, reward, next_state, done))
        state = next_state
        if done:
            break

        if length(replay_memory) > batch_size:
            minibatch = replay_memory.sample(batch_size)
            for s1, a, r, s2, done in minibatch:
                target = r
                if not done:
                    target = (r + gamma * amax(target_agent.predict(s2)))
                targets = target_agent.predict(s1)
                targets[a] = target
                agent.train(s1, targets)

            updated = agent.weights * tau + ((1 - tau) * target_agent.get_weights)
            target_agent.weights = updated
```

An optimal solution will be attempted by grid searching through various parameter values. The following values are intended to be the hypothesis space to search for the optimal value although there may be more values added/removed depending on time constraints.

Parameter	**Value 1**	**Value 2**	**Value 3**
Decay	0.01	0.001	0.0001
Epsilon	0.99	0.9	0.7
Gamma	0.7	0.9	0.99
Learning Rate	0.01	0.001	0.0001
Tau	1.0	0.1	0.01

References

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., & Wierstra, D. (2013). Playing Atari with Deep Reinforcement Learning.

NASA. (2018, December 19). *MARS Insight Mission*. Retrieved from NASA: https://mars.nasa.gov/insight/timeline/overview/

OpenAI. (2018, December 19). *Getting Started with Gym*. Retrieved from OpenAI: https://gym.openai.com/docs/

OpenAI. (2018, December 19). *LunarLanderContinuous-v2*. Retrieved from OpenAI Gym: https://gym.openai.com/envs/LunarLanderContinuous-v2/