Machine Learning Engineer Nanodegree

Capstone Proposal

David Peabody
August 1st, 2017

Implementation of Deep Q-learning on OpenAI's Gym Environment

Abstract

In this project I aim to implement a deep reinforcement learning approach on OpenAI's Gym environments. I intend to use the Atari Pong-v0 as the primary learning environment and if possible scale the same approach through to other Atari environments. Initially I will implement a form of the Q-learning algorithm along with a neural net coded in Keras. The overarching goal of this project will be to provide a clear, step-by-step explanation of how the algorithm works to solidify the knowledge for myself and for others.

Domain Background

Reinforcement learning (RL) is an exiting subject area because it represents the ability to train machines to find solutions (policy's) to complex dynamic problems without hardcoding instructions which could be prohibitively expensive effort wise or even impossible.

Advances in RL could lead to progress in manufacturing [1], robotic control [2], medical research, finance and engineering. Further, research into how agents can learn most efficiently can provide insight into how humans learn or more often how there are other ways to learn beyond how humans traditionally have.

Deep reinforcement learning is an active area of research in the machine learning community with weekly releases of new papers covering innovations including, algorithm design and implementation [3], new reward mechanisms & policies [4],

varying neural network structures, and parameter innovations such as injecting noise into the algorithms parameters [5].

Problem Statement

Recent advances in machine learning has provided techniques to extract high dimensional sensory inputs from sources such as vison (6). The problem investigated in this project is how to use this sensory input to train an agent to select actions in a way that maximizes its future rewards. The agent will only be provided with the same information a human player would have and will not be provided any custom features. The environment the agent will act on in this project will be Atari's Pong-v0. A successful application will be considered to be an agent that acts as well or better than an average human agent.

Datasets and Inputs

Unlike traditional supervised learning, reinforcement learning problems do not come packaged with a labeled dataset in hand. In RL the dataset is built up based on the experiences of the agent. In this project, the input is an RGB image of the video stream, which is an array of shape (210, 160, 3), Each image represents a state. Along with that state we will also track the action taken, the reward received and the corresponding new state. The log of this information is the equivalent of a supervised dataset where the action for a given state and the reward received build the policy of how the agent should act in each situation.

The video stream input and space in which the agent acts is OpenAI's Gym environment, specifically the Atari game Pong-v0. OpenAI Gym is a toolkit for reinforcement learning research. It includes a growing collection of benchmark problems that expose a common interface, and a website where people can share their results and compare the performance of algorithms [7].

The two key aspects of the Gym environments are:

1) Providing an easy to use, consistent interface so that a single algorithm design can be tested against a wide variety of environments with few to no alterations. Thus, illuminating the strengths and weaknesses of an algorithm in various situations.
2) Implementing strict versioning requirements to the environments so that the relative capabilities of algorithms can be compared against each other, apples to apples for the foreseeable suture.

On a final note, the RGB image from the video stream may have to go through a pre-processing stage to make it lighter to train on and easier for the neural network to gather useful information on. My intention is to do the minimal preprocessing possible as I would like the algorithm to be able to work effectively in as many environments as possible, and the more the image input is pre-processed the less this will be true. Options for preprocessing include:

- Reducing the size of the input
- Converting the image to greyscale
- Subtracting the past frame from the current frame, so all that is left is the difference or what has changed i.e. movement.

Solution Statement

The solution implemented in this project will be a basic deep Q-learning algorithm. The deep Q-learning algorithm combines the visual processing and learning capabilities of a deep neural network with an implementation of the Q-learning algorithm (shown below).

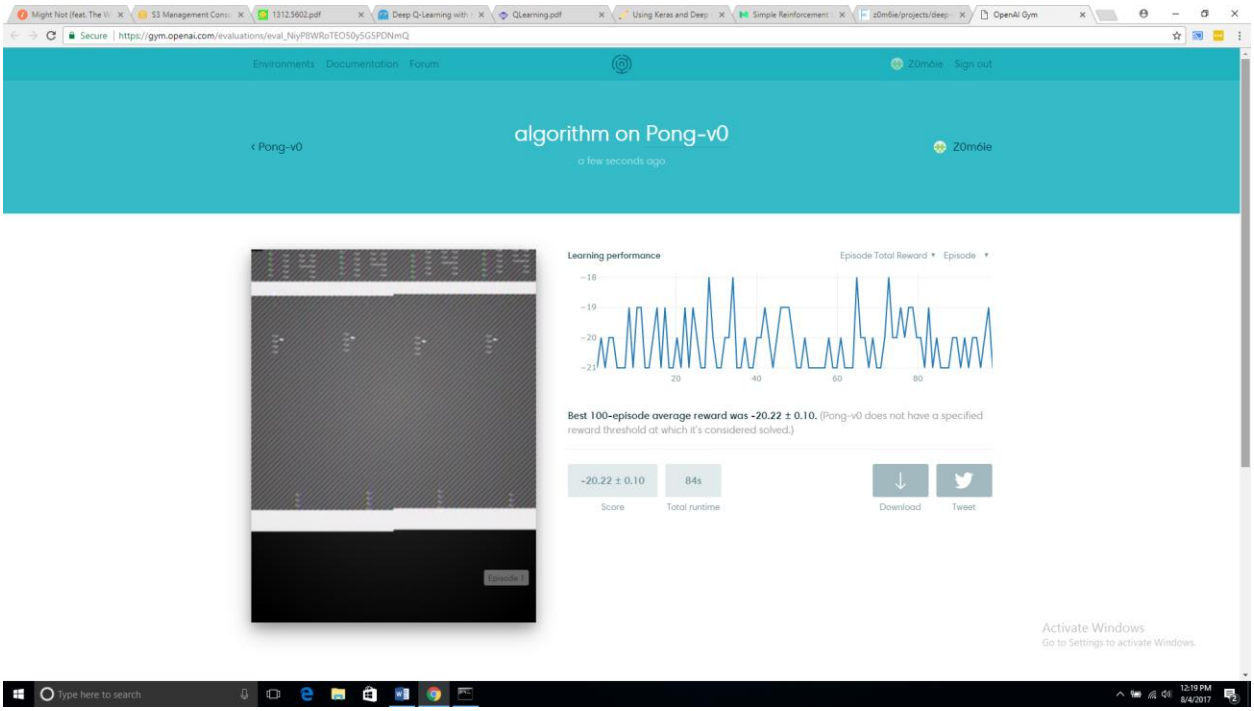
Equation 1) $$Q(s,a) = r(s,a) + \gamma(\max(Q(s',a')))$$

Where the aim at each state (s) is to choose the action (a) which maximizes the utility function, Q(s,a). In the equation above, the utility function, Q(s,a), is the estimate of how beneficial a given action is from a specific state. This is calculated by taking into account the immediate reward, r(s,a), plus the discounted (γ) maximum reward for the future state (s').

As this solution is tried and tested, my focus for this project will be to not only successfully implement a deep Q-learning solution but to provide a clear explanation of the implementation to solidify my own knowledge but hopefully to provide a clear guide for other students just getting to grips with deep reinforcement learning.

Benchmark Model

Throughout this project I shall be using OpenAI's scoring platform which evaluates the upload of your model's performance over a 100-episode test case the average.



For the proposal, I will provide two benchmarks which I will compare my learning algorithms against throughout the project.

The first is an implementation of a random agent, this agent randomly reward was **-20.22 ± 0.10** for more information and a video clip please follow this link. N.B. The video currently shows four copies of the same visual output. This is due to the program being run on a remote server without a functional monitor. This will not effect the training on the learning algorithm.

My second benchmark is the score I can achieve as a human agent over a 5-game stretch. This provides a decent baseline as to whether my learning algorithm plays better than an average human. The human agent score achieved is **-9**. Apparently, I am no good at pong either.

Evaluation Metrics

As mentioned in the above section, the model will be evaluated against OpenAI's platform. The gym platform takes the average reward received for a the best 100-episode period. The gym platform is designed specifically for reinforcement learning hence why algorithms are compare on the average rewards received. In addition, as a further method we will track the number of episodes required to produce an effective model.

Pong ends when one player scores 21 points, so the average reward metric is the average reward received in a series of 21-point games. To be more explicit, I have created a table to emphasize what the possible reward scores mean in relation to the game.

Reward	Example Score		Note
	CPU	Agent	
-21	21	0	Agent never scores and CPU reaches max score. All rewards received are negative
0	21	21	Using a 2-game span a reward of 0 represents an agent that is equally matched to the CPU. I.e. they score an equal amount of points to each other
+21	0	21	This is the ultimate agent it scores 21 points without letting the CPU score ever.

Project Design

The initial implementation for the project design will be a simple deep Q-learning algorithm (pseudo code below). The intent will to achieve a learning algorithm, and then to improve on that algorithm either with parameter tuning or by considering more complete implementations and possibly other deep reinforcement learning algorithms.

In addition, if time permits I will run the algorithm against other environments on the OpenAI platform.

The following pseudo code outlines the key steps of the algorithm

Pseudo Code for calling the agent:

- Make environment
- for i_episodes in range(number of training episodes)
    - state = env.reset()
    - while True:

        - action = agent choose action
        - new_state, reward, done, info = env.step(action)
        - store in list (state, action, reward, new_state, done)
        - make state = new_state
        - if done:
            - print("Episode finished)
            - break
    - Carry out mini-batch update on neural network

Pseudo Code for agent choose action:

- Neural network predicts rewards for possible actions based on state
- Action = Max(rewards for possible actions)

Pseudo Code for mini-batch update:

- Sample a mini-batch from the stored state, action, reward, new_state list
- Calculate the expected future reward
- Train the model on the stored state, with the addition of the utility value

The results of the project along with the write-up will also be uploaded to OpenAI's scoring platform. This will provide a Q-learning example for future students and hopefully help pass on knowledge.

References

[1] Mahadevan, S 1998. Optimizing Production Manufacturing using Reinforcement Learning. Michigan, USA. Proceedings of the Eleventh International FLAIRS Conference

[2] Australian Center for Robotic Vision, 2017, https://www.roboticvision.org/

[3] Mnih, Volodymyr 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG]

[4] Christiano, Paul 2017. Deep reinforcement learning from human preferences. arXiv:1706.03741 [stat.ML]

[5] Plappert, Matthias 2017. Parameter Space Noise for Exploration. arXiv:1706.01905 [cs.LG]

[6] Mnih, Volodymyr 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602v1 [cs.LG]

[7] Brockman, Greg 2016. OpenAI Gym. arXiv:1606.01540 [cs.LG]