Using Reinforcement Learning on an Inverted Pendulum Game

Andrew Sylvester and William Zhou

July 2019

Abstract

This paper describes a neural network designed to interact in a simple game environment. The network, deep feed forward, combines deep reinforcement learning with the Adam optimizer and applies this to an inverted pendulum game. This paper describes the importance of neural networks, the process in which we created a deep feed forward network, and the strengths and weaknesses of the AI along with the results.

Introduction

To demonstrate our understanding of neural networks, we established a deep feed forward neural network. Deep reinforcement learning is incorporated in order to train the neural network to "play" a game. Notably, this type of learning is significant in its ability to learn through past experiences, almost exhibiting sentient-like behaviors. Intermediate levels of knowledge are required in Python and multiple packages, as well as a wide depth of experience in machine learning. Neural networks are not immensely new to the field of Artificial Intelligence and as a result, many talented researchers have created neural networks that surpass the complexity of our experiment. It is important to note that the presented experiment is of elementary difficulty and is only a simple stepping stone for individuals interested in AI.

Methods

We used the gym package developed by Open AI for our environment. Open AI has multiple different standalone environments and also has the option to use either MuJoCo (multi-joint dynamics with contact) for three-dimensional physics environments or ALE (arcade learning environment) for classic Atari games. In addition to gym, we also used the Keras package to implement a deep neural network with two 24-node hidden layers. The success of the AI was measured by how many time steps the cart was onscreen and the pole was relatively upright. We used the Adam optimization algorithm to determine how we modify the weights. We started off by using some example code written by Keon Kim that implemented a deep Q-learning neural network.

$$For\ each\ Parameter\ w^j$$
$$(j\ subscript\ dropped\ for\ clarity)$$

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

$\eta : Initial\ Learning\ rate$
$g_t : Gradient\ at\ time\ t\ along\ \omega^j$
$\nu_t : Exponential\ Average\ of\ gradients\ along\ \omega_j$
$s_t : Exponential\ Average\ of\ squares\ of\ gradients\ along\ \omega_j$
$\beta_1, \beta_2 : Hyperparameters$

Figure 1: The equations for the Adam optimizer

Results

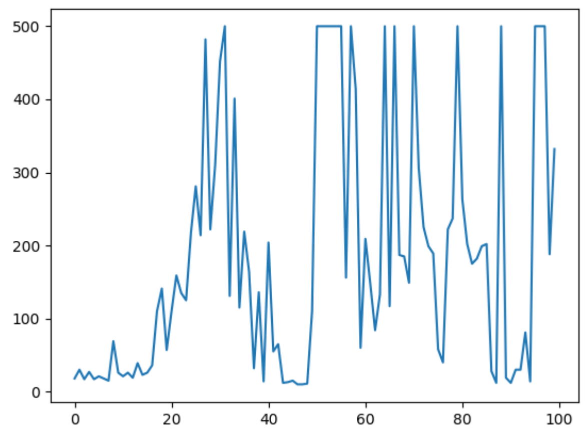

Figure 2: A graph of the results after 100 episodes

Conclusion

These results are somewhat unexpected. The fact that the AI spikes in performance erratically is strange however we believe that this is because our AI implements a form of stochastic gradient descent. Greg Surma, a writer for the website Towards Data Science, has achieved similar results to ours in his article ‘Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning)’. His graph reflects the same highs and lows on performance that ours has:

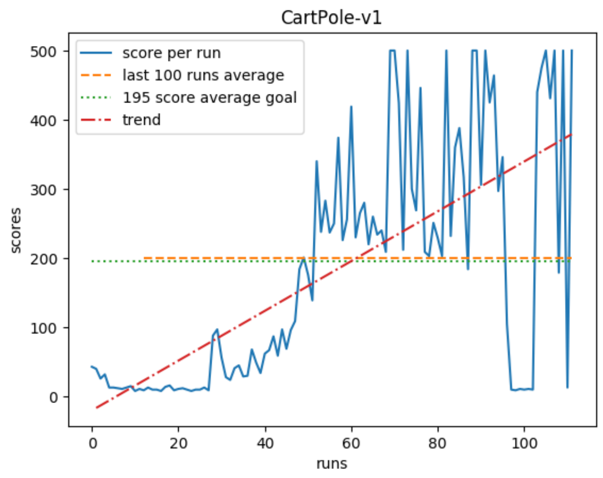

Figure 3: Greg Surma’s AI on the CartPole-v1 environment

Other people have reported getting scores as high as one million although we aren’t entirely sure if those are real scores or just made up ones. Given more time we would like to apply the same algorithms to other games in the Open AI gym package. We would like to try out the rest of their ‘Classic Control’ environments and then move on to more complex environments like the MuJoCo environments and the Atari environments.

References

https://gym.openai.com

https://keras.io/

https://keon.io/deep-q-learning/

https://towardsdatascience.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288