**Training the BipedalWalker with the OpenAI Platform**

Jing Cui, Zhenying Tao and Yunchuan Kong

**Problem and Motivation**

As mentioned in our initial project proposal, walker problems have been extensively studied in the field of artificial intelligence. Learning from the case “CartPole” demonstrated in class, we obtained some basic knowledge of training an agent whose behavior is defined by actions of joints. Also, through the class project “Pong” game, we were eventually able to implement algorithms with the OpenAI platform, though having encountered challenges when setting up OpenAI environment on Windows systems at the beginning. At the same time, we gained experience working on useful Python packages such as Tensorflow and Neurolab.

Usual reinforcement learning algorithms such as Qlearning work merely fine for relatively simple problems such as CartPole. However, things can become more complicated when the number of joints increases. Therefore, in this project, we are trying to train a more advanced agent, namely the “BipedalWalker” provided by OpenAI. The primary goal is to let the agent be able to walk forward with a reasonable speed.

**Approach**

Inspired by our implementations for the Pong game, we mainly consider employing neural network approaches for training the bipedal walker. We utilize two main algorithms, the ordinary deep neural network algorithm and its variation, the genetic algorithm. An ordinary deep neural network model contains an input layer, an output layer and hidden layers, while the number of hidden layers can vary. The term “deep” is thus due to the multiple layers of the hidden structure. The genetic algorithm has a similar model structure to the deep neural network, however, it includes a mutation mechanism to change the number of nodes in the hidden layers.

**Implementation**

**Ordinary Neural Network Algorithm**

The ordinary deep neural network algorithm is implemented with the Python package Neurolab. In the BipedalWalker environment, the input (observation) is a twenty-four dimensional vector, and the output (action) is a four dimensional vector.[[1]](#footnote-1) Thus, the input and output layers in the neural network model should contain twenty-four and four nodes respectively. Given these relatively small numbers and the small difference of numbers between the input and output layers, it is not likely to be appropriate to use more than two hidden layers. Consequently, our experiments are based on neural networks with one and two hidden layers.

Besides choosing the number of hidden layers of the network, discretizing of the input and output is also of our consideration. Discretizing the observation and action spaces is able to transfer the continuous control problem into a discrete one, potentially making the problem easier to handle. In our implementations with Neurolab, both discretized and original observations are experimented.

**Genetic Algorithm**

**Results**

**Ordinary Neural Network Algorithm**

Unfortunately, the ordinary neural network model implemented by Neurolab did not result in any good result. Table 1 summarizes our main results after over 5000 training episodes, where the entries are the corresponding best scores obtained.[[2]](#footnote-2)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Continuous input/output | Discretized input/output | Partially discretized input/output |
| Hidden layer1 with 16 nodes & hidden layer2 with 8 nodes | -101.53 | -99.82 | -100.77 |
| Hidden layer with 16 nodes | -101.75 | -99.96 | -101.34 |

Table 1: results of the ordinary neural network algorithm

From the table, although it seems that discretizing is a bit helpful, it is still far from achieving an acceptable performance. On the contrary, introducing one additional hidden layer does not seem to contribute at all, as the results are rather similar compared to those with only one hidden layer.

**Genetic Algorithm**

**Discussion**

1. Details about the environment can be found at https://gym.openai.com/envs/BipedalWalker-v2. [↑](#footnote-ref-1)
2. The BipedalWalker is regarded as solved when consistent scores of over 300 are obtained. [↑](#footnote-ref-2)