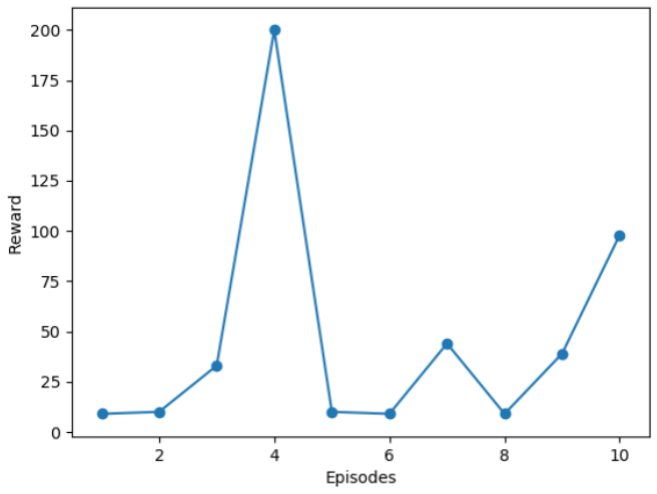
Exercise 2a

In this exercise, we need to run “CartPole-v0” agent for 10 episodes each lasting 200 steps. After importing the required libraries, first we make “CartPole-v0” environment. Then we observe length of vectors in observation and action space. The action for cart pendulum is either +1 or -1 indicating either move the cart towards right or left.

For each episode, environment is reset and training parameters are initialized randomly using a Gaussian distribution with average 0.0 and variance 0.1. In this case, we are making a 4-5-2 topolgy policy network which means there are 5 hidden neurons and 2 neurons in the output layer. Hence our training parameters are weight matrix between input layer and hidden layer (W1), bias for hidden layer (b1), weight matrix between hidden layer and output layer (W2) and bias for output layer (b2). The reward for this episode is initialized to zero and after that we enter into the loop for running steps.

The policy network is solved and action is taken as maximum among the two output values. Then the agent this this action and returns new state, reward, done and info. The reward is added into reward for this episode and state is updated. Finally we put reward for each episode into a list and plot rewards vs episode.



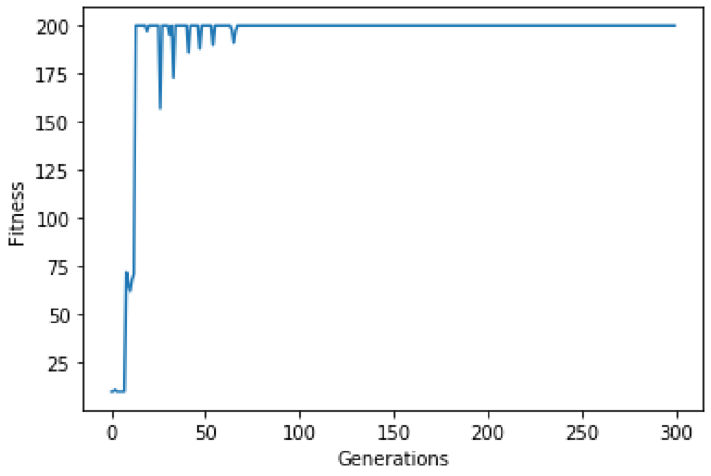
Hence, the agent did not learna anything over the episodes. Every time the code is run result would be different as well due to random initializations.

Exercise 2b

In this exercise, we will adopt evolutionary strategy and train our agent over the generations. In each generation, we have two individuals. Each of these individuals will run for an episode and its reward will be collected. The individual with higher reward will continue to to the next generation but the individual with lower reward will be replaced by a normal distribution. Hence in this way, we are keeping the original information as well as trying new individuals in the population over the generations.

First of all we intiliaze random weights and biases. A populations is made consisting of 2 individuals and each individual has 37 total parameters (weights + biases). We run a loop for generation and for every generation, we loop through each individual. We reshape the individual vector into weights and biases and run an episode.

From my experiments trying several values, the best result I got is shown below:



The agent has been trained for 300 generations and fitness score of best individual in each generation is plotted along y-axis. We can see that agent quickly trains itself within first 100 generations and after that fitness score remains constant. This is the maximum score I observed by trying different values.

Exercise 3

I saw the acrobot.ini file and found that the environment being used is "**Acrobot-v1**". Acrobot is a 2-link pendulum with only the second joint actuated. Initially, both links point downwards. The goal is to swing the end-effector at a height at least the length of one link above the base. Both links can swing freely and can pass by each other, i.e., they don't collide when they have the same angle.

Observation or state Vector

The observation vector consists of 6 real numbers. It consists of the sin() and cos() of the two rotational joint angles and the joint angular velocities : [cos(theta1) sin(theta1) cos(theta2) sin(theta2) thetaDot1 thetaDot2]. For the first link, an angle of 0 corresponds to the link pointing downwards. The angle of the second link is relative to the angle of the first link. An angle of 0 corresponds to having the same angle between the two links. A state of [1, 0, 1, 0, ..., ...] means that both links point downwards.

Action Vector

The action vector consists of 3 real numbers. It is either applying +1, 0 or -1 torque on the joint between the two pendulum links.

Experiments:

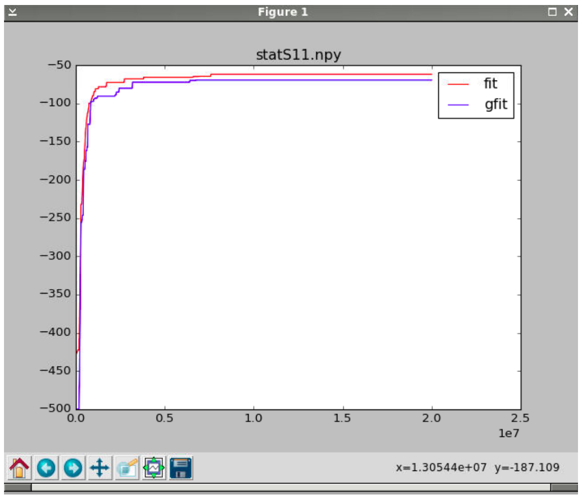
The experiment is run for a seed value of 11 using the following command:

- *python3 ../bin/es.py -f acrobot.ini -s 11*

It took 2hours and 10 minutes for complete training and within this time best fit and average fit was always found to be zero. After the training when I plotted stat using the following command

- *python3 ../bin/plotstat.py stats11.npy*

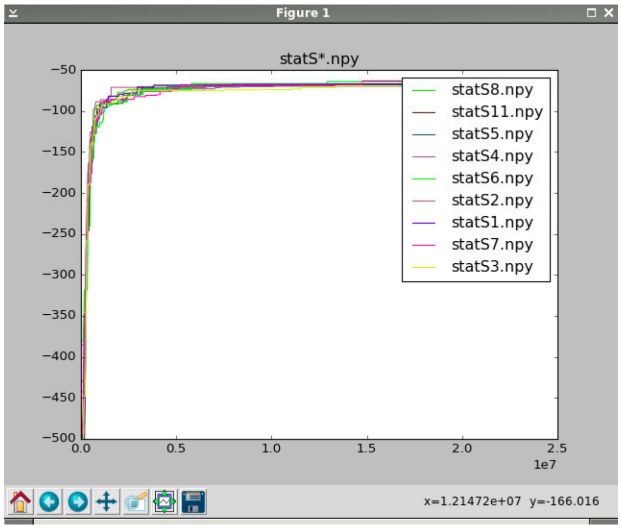
The following result is obtained.



In order to compare my training results with the results available on Gothub repository, I entered the following command

- *python3 ../bin/plotstat.py*

The following result is obtained:



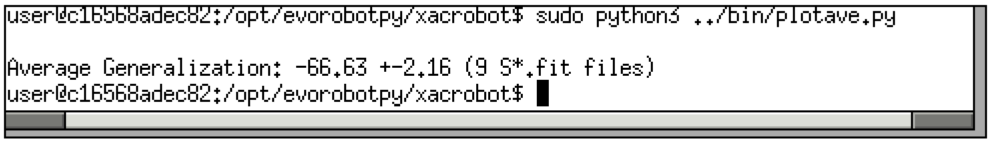
The comparison suggests that seed has very small effect and more or less, training is really good for all seeds. I have tried the following command as well.

- *python3 ../bin/es.py -f acrobot.ini -t bestgS11.npy*

This command shows live demo of working of the agent trained. I tried the following command:

- *python3 ../bin/plotave.py*

Following result is obtained:



```
user@c16568adec82:/opt/evorobotpy/xacrobot$ sudo python3 ../bin/plotave.py

Average Generalization: -66.63 +-2.16 (9 S*.fit files)
user@c16568adec82:/opt/evorobotpy/xacrobot$
```