# **Task 1**

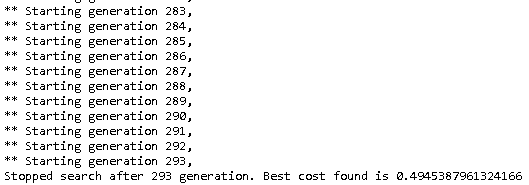


Figure 1-1. Result of Best Cost

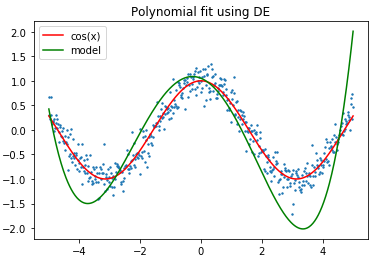


Figure1-2. Polynomial fit using DE

The above pictures show one of our results using differential evolution to fit a polynomial noisy dataset.

Since we decide the target cost is 0.5, the algorithm finds a cost which less than 0.5 at the 293 generation. Therefore, the suitable w values are found to build model to fit cos(x).

# **Task 2**

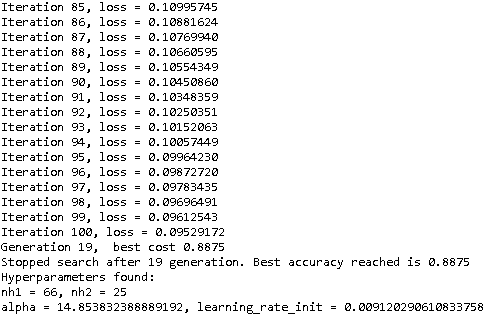


Figure 2-1. Result of Best Cost

Through differential evolution algorithm, we get the best accuracy is 0.8875. For this neural network, its first layer has 66 neurons and its second layer has 25 neurons. Another hyper parameters alpha and learning rate is demonstrated in Figure2-1. In task 2, we set if the accuracy is bigger than 0.9, the program will stop and return the result. However, after 20 generating process, there are no accuracy that is bigger than 0.9. The main reasons could be 0.9 is a high accuracy, which is hard to reach randomly with 20 maxiter generations.

# **Task 3**

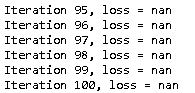
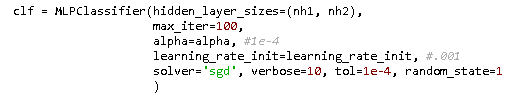


Figure 3-1. Nan values

During the process of Task 2, there are some Nan of loss value. It results from that the loss of value is too big, which causes the data overflow. Because the w is selected randomly at the beginning, it may lead the root mean squared error too large. In the Task2, it uses to build neural networks. The code is shown blew:



Since the code does not define the activation parameter, the default activation function is ‘relu’. The ‘relu’ function return f(x) = max (0, x), so there may be extremely large x that causes the data overflow.

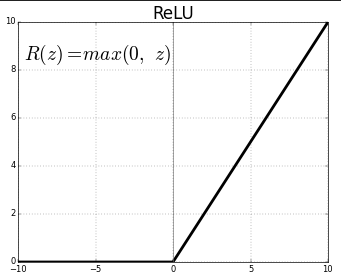


Figure 3-2. ReLU function

To solve the Nan value problem, we have just changed the ‘relu’ function to ‘tanh’ function, which returns f(x) = tanh(x). It make the x is located between -1 to 1(show in below picture).

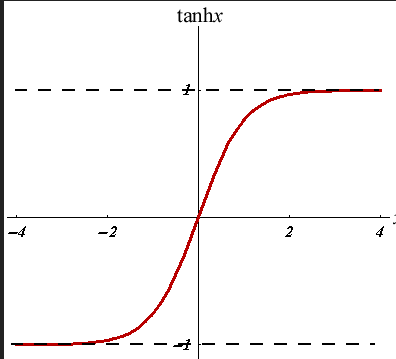


Figure 3-3. Tanh function

Thereby, in task 3, the activation function “tanh” is utilized to handle the Nan value problem.

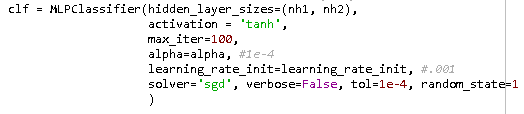


Figure 3-4. Tanh activation

In Task 3, we need to compare the [(5, 40), (10, 20), (20, 10), (40, 5)], which is four computational budget to train neural network. To decide which structure is better, the test accuracy is selected to compare the ability for neural network. However, according to the experimental results, all four kinds neural network cannot reach the accuracy of 0.9. Hence, to differentiate the capability of four allocations, we have changed the expected accuracy to 0.885. Then we run the task3() 10 times to get the results (Shown in the below picture).

The below picture is detail information for 4 strategies in one time:

Figure 3-5. Result of 4 allocations

The below picture shows the best accuracy in running 10 times:

Figure 3-6. Result of Best cost

According the average accuracy results, we discover (20, 10) strategy has the best performance.

Table 3-1. Average Accuracy

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Time 1 | Time 2 | Time 3 | Time 4 | Time 5 | Time 6 | Time 7 | Time 8 | Time 9 | Time 10 | Average accuracy |
| (5,40) | 0.875 | 0.88 | 0.875 | 0.875 | 0.8875 | 0.8875 | 0.87 | 0.8825 | 0.875 | 0.8725 | 0.878 |
| (10,20) | 0.885 | 0.8775 | 0.88 | 0.8775 | 0.8825 | 0.875 | 0.87 | 0.88 | 0.8825 | 0.8775 | 0.87875 |
| (20,10) | 0.87 | 0.875 | 0.8775 | 0.87 | 0.87 | 0.875 | 0.875 | 0.875 | 0.875 | 0.885 | 0.87475 |
| (40,5) | 0.865 | 0.8725 | 0.8875 | 0.88 | 0.8725 | 0.8725 | 0.8725 | 0.8725 | 0.8675 | 0.8775 | 0.874 |