# Regression model MLP design

**Training data:**

|  |  |
| --- | --- |
| X | 200 equally spaced points between -2 and 2, generated using torch.linspace function. |
| Y | *y* = *x*3 + 2*x*2 + 0.3*b* , where *b* is a random number drawing from a normal distribution with zero mean and variance of one, generated using torch.randn function. |

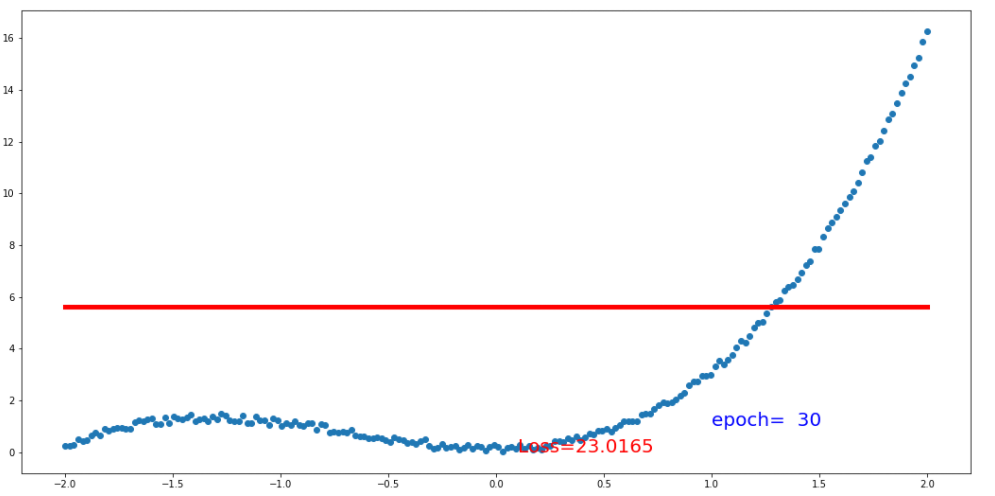
**MLP NN design:**

|  |  |
| --- | --- |
| Hidden layers | The MLP should contain 3 hidden layers with 5, 10, 8 neurons, respectively. |
| Activation function | Use the same activation function for all neurons in the same layers. Let the activation function for the three hidden layers be: relu, sigmod, and relu, respectively. |
| Loss function | Use mean square error to calculate the loss. |
| Optimization function | Use SGD for problem 1 and gradient decent method for problem 2. |
| Learning rate | 1 |

**Training design:**

|  |  |
| --- | --- |
| Epoch | Train the MLP 30 epochs. |

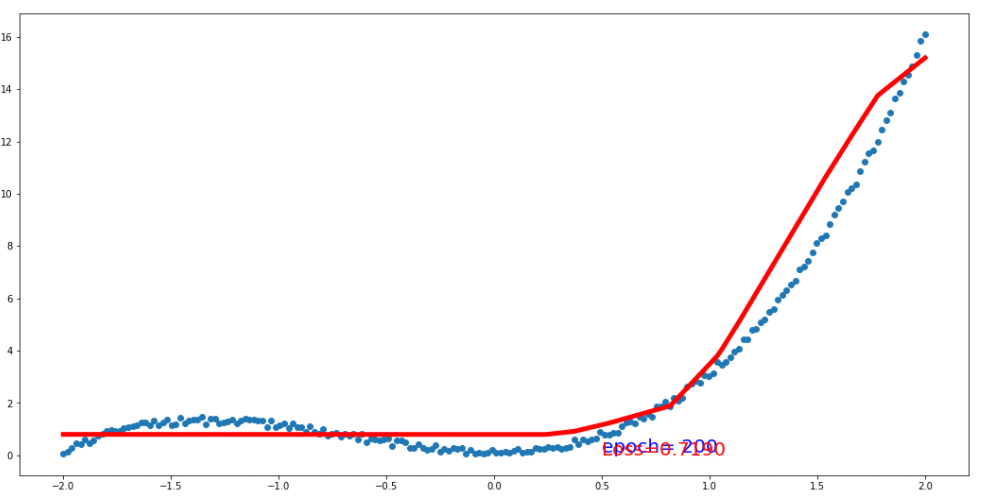
The network is not learning correctly and the error is high.



# Anthony Spence (施東尼)

1. In order to improve the first exercise, the following steps were taken:

* Change the activation function of the hidden layer so that all made use of relu. The basic idea is that relu considers all the values independent of the size of the values, sigmoid on the other hand ignore smaller values. After doing this the error improve slightly by 0.5.
* The next step was to reduce the learning rate from 0.5 to 0.02. Reducing the learning rate avoid the neural network to get stuck in local minimum. By taking this step the loss function was reduced from 15 to a value of 4.
* The final step taken in order to improve the result was increasing the number of epoch from 30 to 200. A higher number of epoch allows the neural network more time to update the weights and thus reduce the loss function, in simple words the NN learns more effectively. By takin this step the result was improved from 4 to 0.7 approximately. The final result can be seen below.

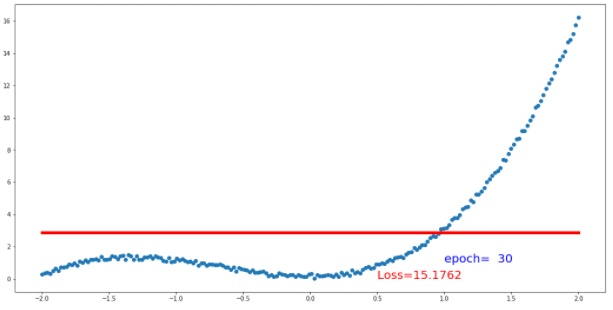


# Hoang Ha (洪航)

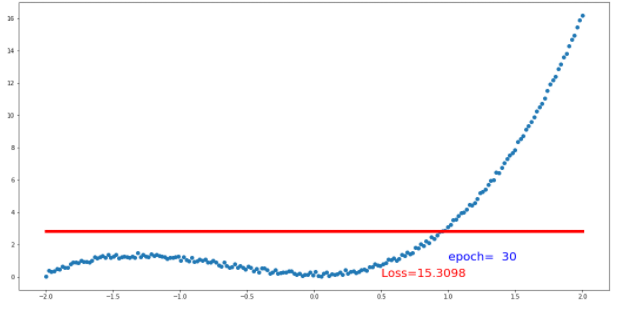
We have the parameter to change in here

* Number of each layer
* Activation function
* Learning rate
* Optimization function
* Epoch

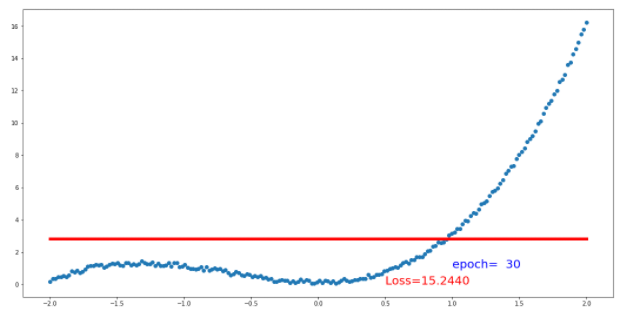
1st : I try to change the learning rate to 0.3 but it won’t work , the result is still the same as previous version



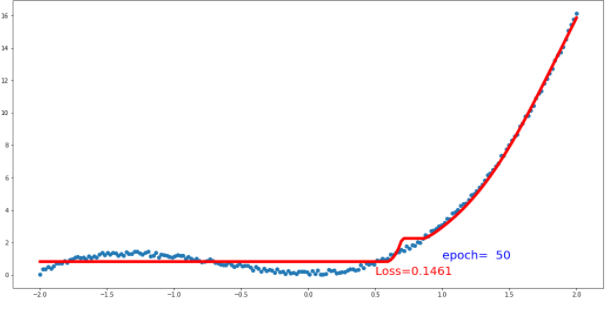
2nd , keep all the same, I try to change the activation function from relu, sigmoid, relu. To relu, relu, relu, but it won’t work as well.



3rd: I add one more hidden layer, but still, it won’t work.



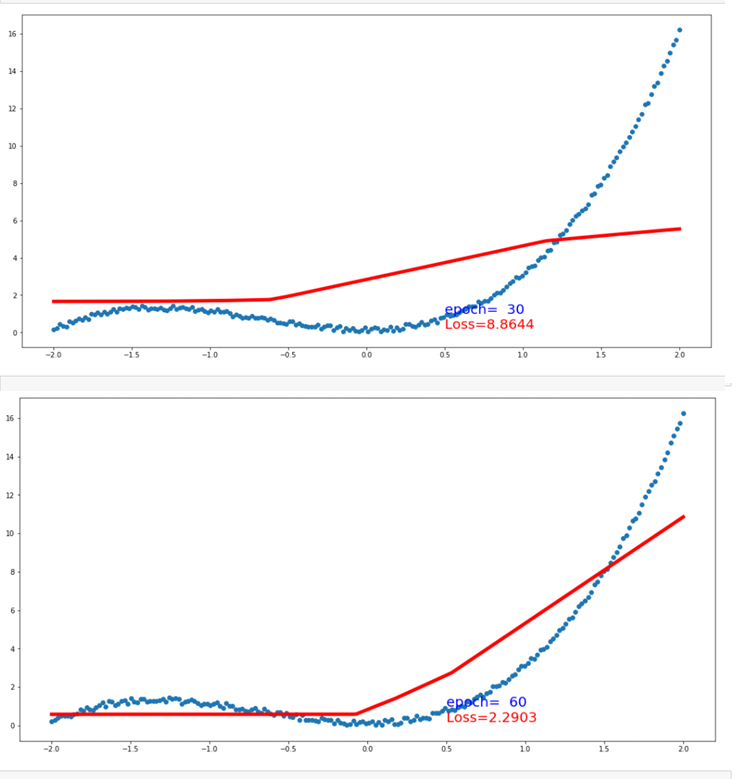
4th: I change to Adam in optimization function, 0.3 learning rate and get the best result



# 余梅粉 (Vanny Minanda)

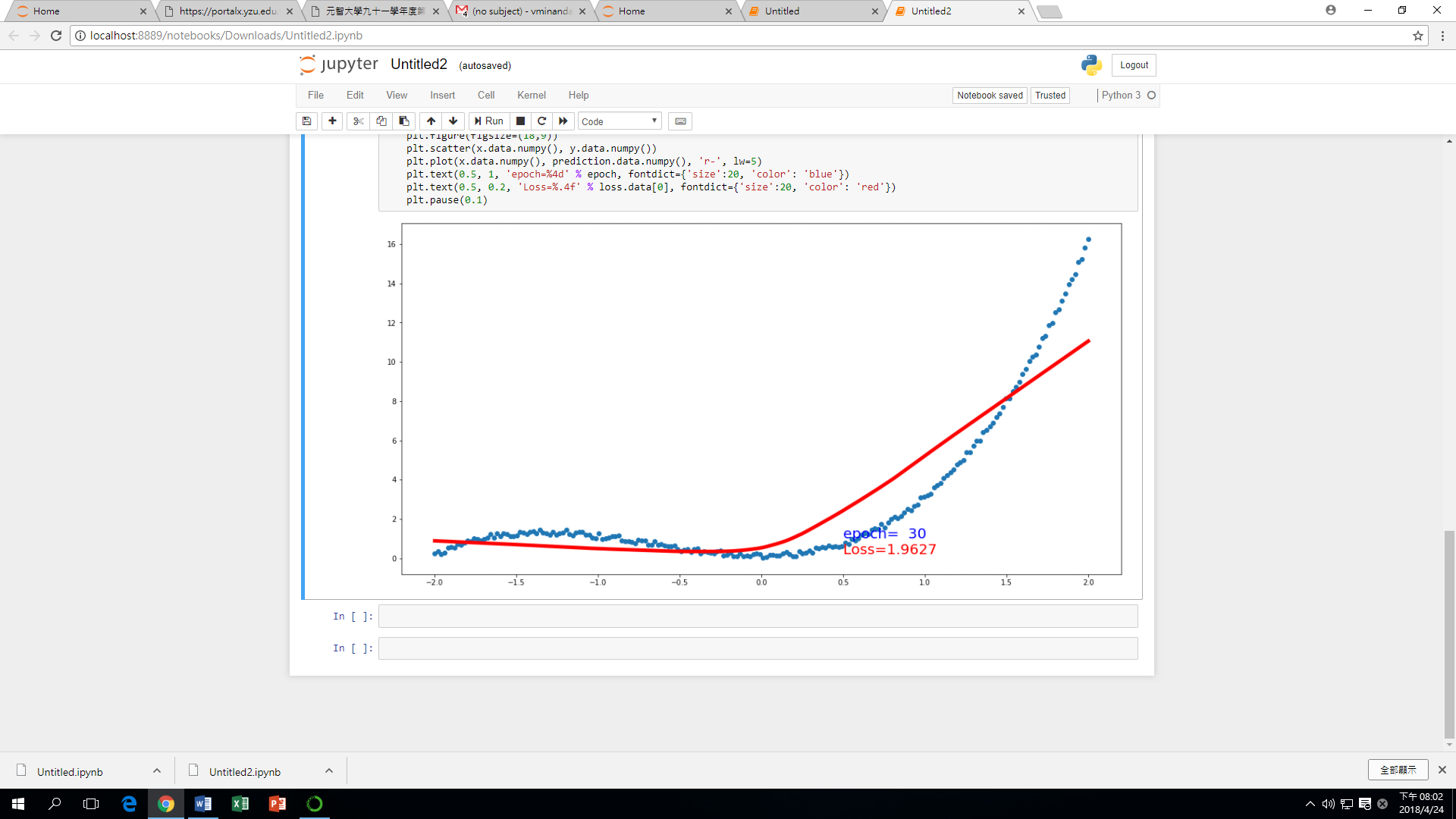
1. Changing the number of epoch :

By increasing the number of epoch become 60 I got better solution if compare with train the model with 30 epochs. This is seems like the model learn eventually by adding more epochs. From picture below, shown that the loss function is eventually decreasing from 8.8644 with 30 epochs become 2.293 with 60 epochs.



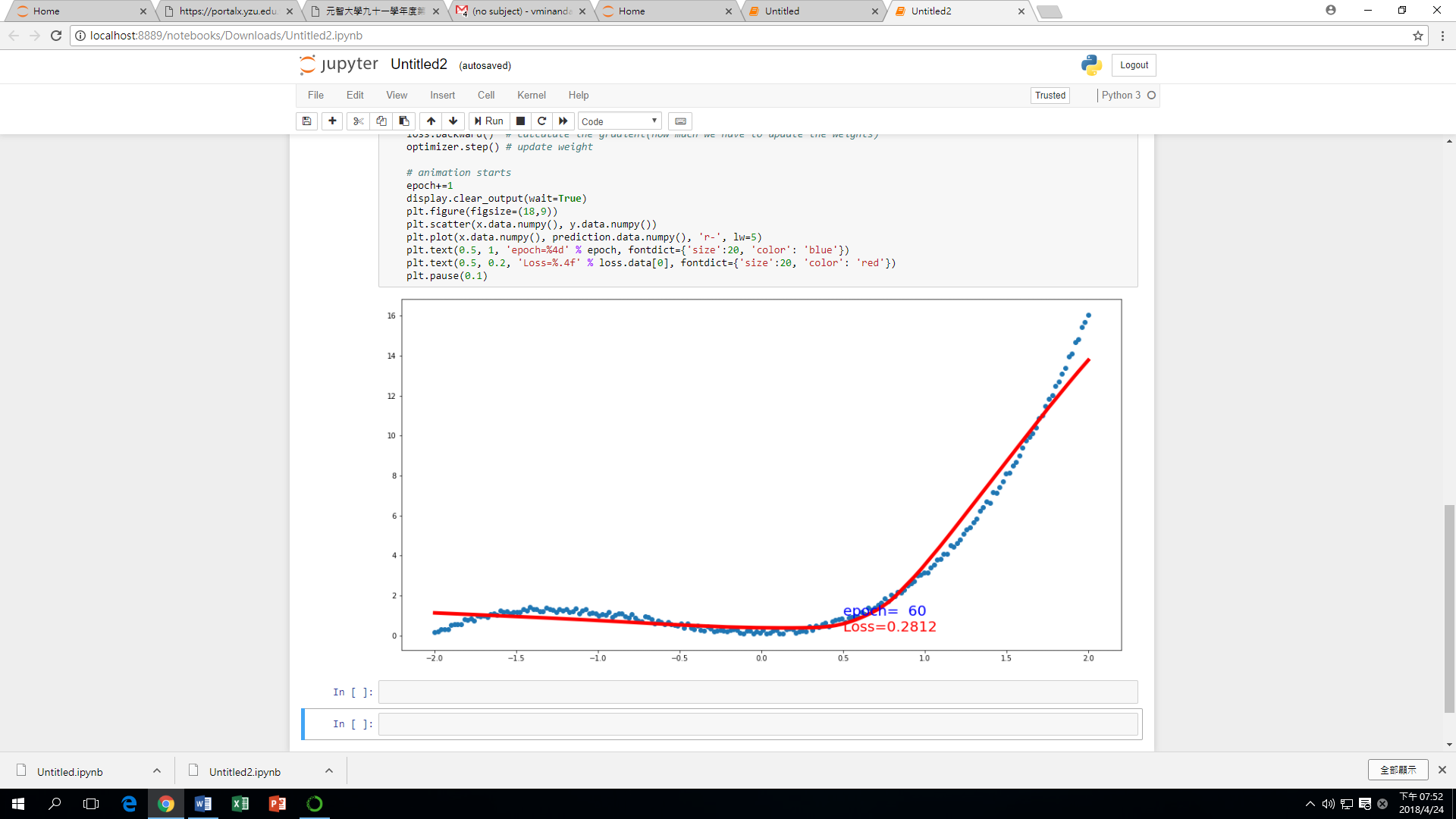
1. Adding the number of layers

By adding the number of layer from 5,10,8 neurons respectively to hidden 1, hidden 2 and hidden 3 to 100, 110, 110 the better result with loss function 1.9627 is obtained



1. Changing the learning rate

By adding the amount of learning rate, the solution will obtained faster, but it doesn’t guarantee to get the best solution. But, if the learning rate is to small, to achieve the best solution required too much time, that’s why defining the learning rate is so important. The experiment held by setting the learning rate equal to 0.01 and the model proof to learn better with the loss equal to 0.2812

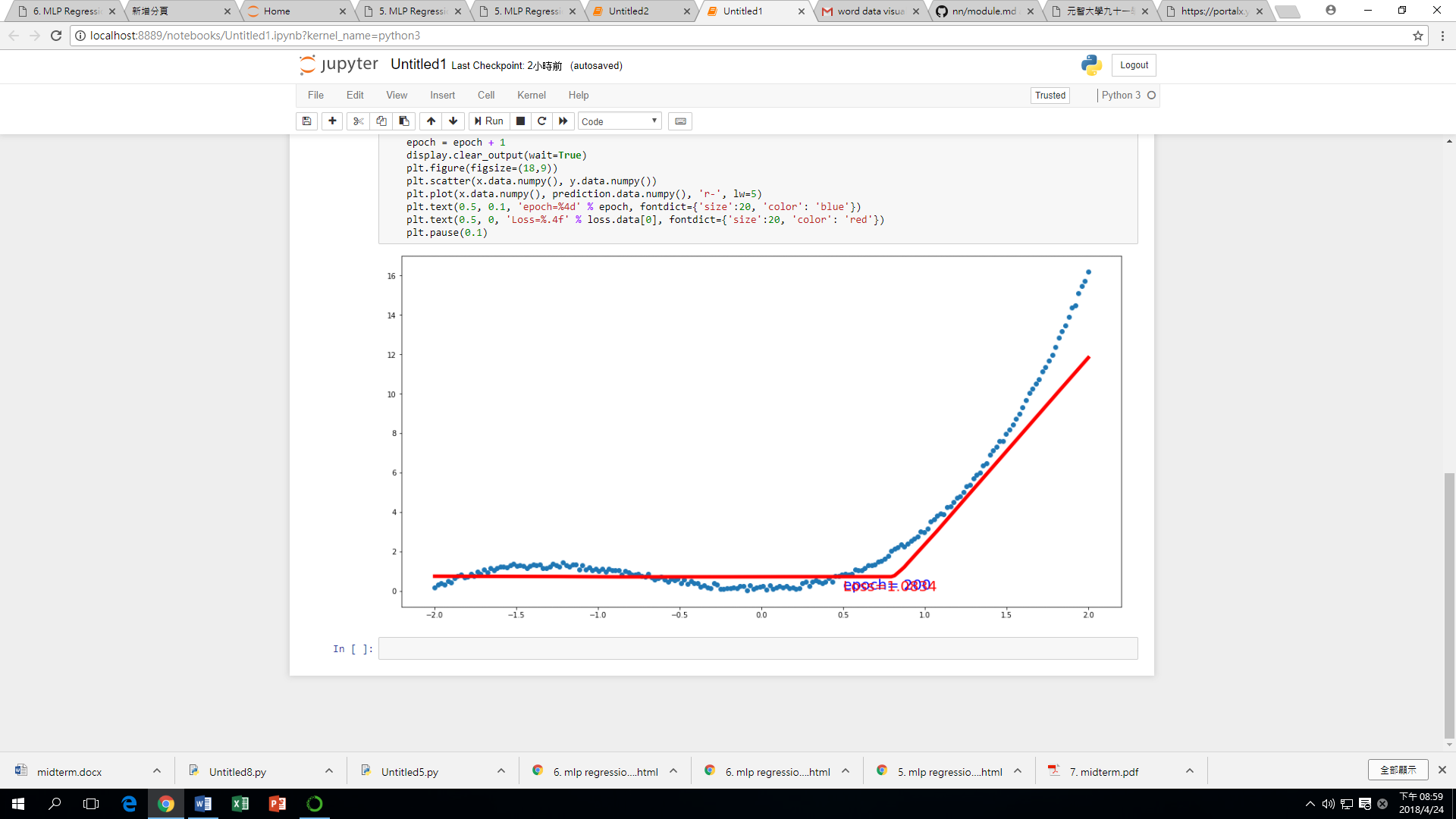


# 賈格蓓 (Gabriela Travisany)

**In order to improve the NN of exercise 1 the parameters that I changed were the following:**

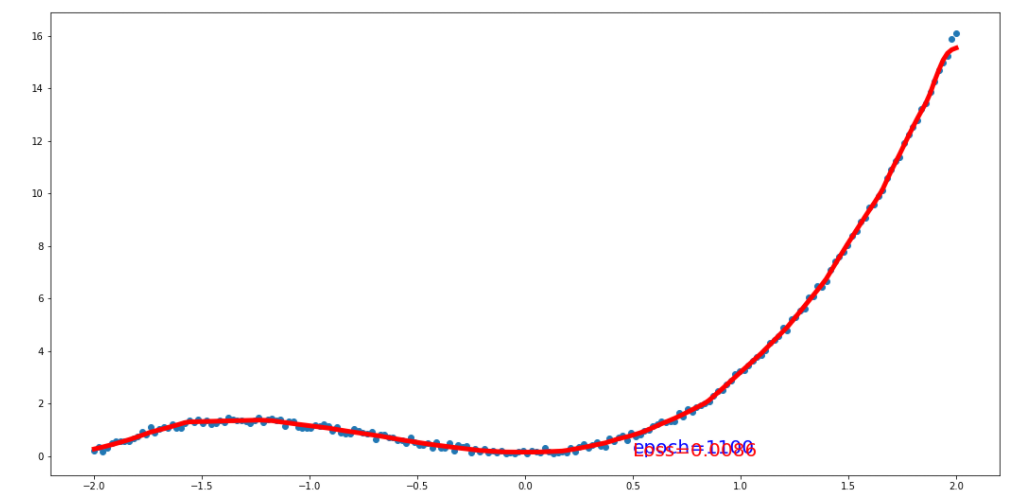
1. Used a learning rate of 0.02, this will allow that the NN evaluate more points and will have more information to learn.
2. Increased the number of epoch from 30 to 100
3. Defined the activation function with Relu, because maybe the previous ones that were using sigmoid at the moment of the update destroyed the small values or large values of the gradients. On the contrary relu allow to save all the values.

With this changes I could make that the NN decrease from 15 to 1. And follow the shape of the equation. See appendix Exercise 3



# 徐培銘

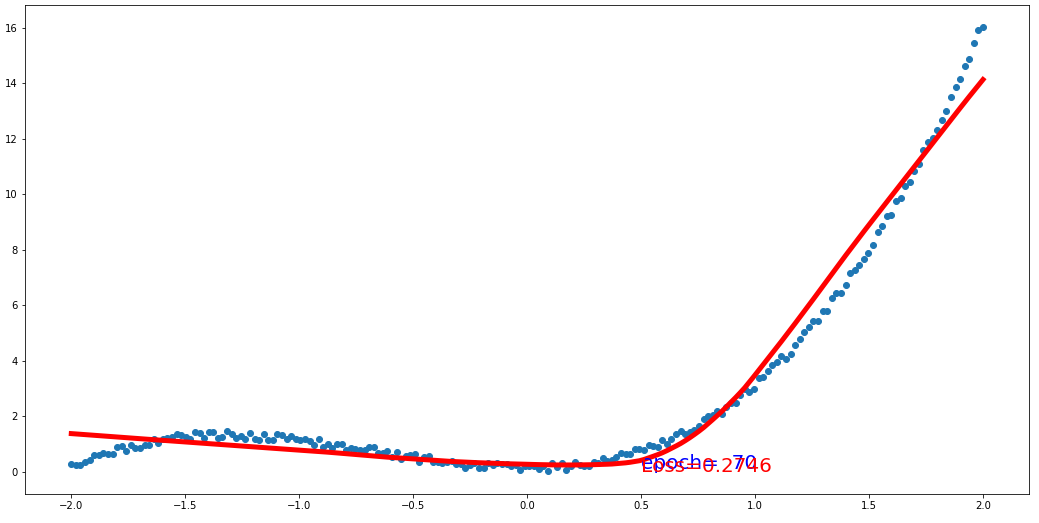
增加隱藏層至四層，一至三層激活函數使用relu，第四層使用sigmoid，嘗試過使用softmax但是得到的結果並不好。修改使用neurons數量為32,64,64,32，並將優化函數修改為Adam，最後將learning rate調至0.01使其不會太大幅的變動並增加其迭代數至1100讓他慢慢接近目標



# Tomas Mendoza (湯馬司)

ERROR = 0.27

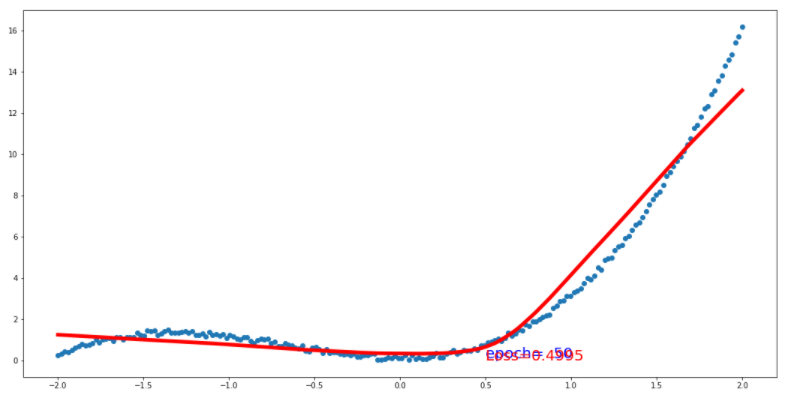
My debug process was based on the fact that Sigmoid activation function causes the vanishing gradient problem, I knew I had to get rid of it and use only Rectilinear activation function (RELU). Also, I noticed that even though my training was improving over time, there was still improvement to be made so I decided to increase my number of neurons in each layer. Finally, I decided to try with different epochs numbers to try and find the optimal value which was 70 epochs.



# DANIEL MOLINA (蒙丹尼)

To improve the NN, I mainly changed the number of nodes in each layer. Using 128 nodes in the first layer, 128 nodes for the second layer, and 64 nodes for the third layer. Leaving the input and output with 1 node.

I lowered the value of the learning rate to 0.1 and used F.Relu function to do the pass. The error loss went down to 0.4995.



# Yona Maimury (鍾如娜)

After building the neural network using 2 different methods. In the problem 3, we try to modify several parameter and the optimizer function, to expect better results (model refinement)

Figure 9 shows the result of changing several parameters including:

1. Number of epoch: from 30 to 100
2. Change the learning rate from 1 to 0.2
3. Change the optimizer function from SGD to Adam. Adam optimization algorithm is an extension to stochastic gradient descent procedure to update network weights iterative based in training data.

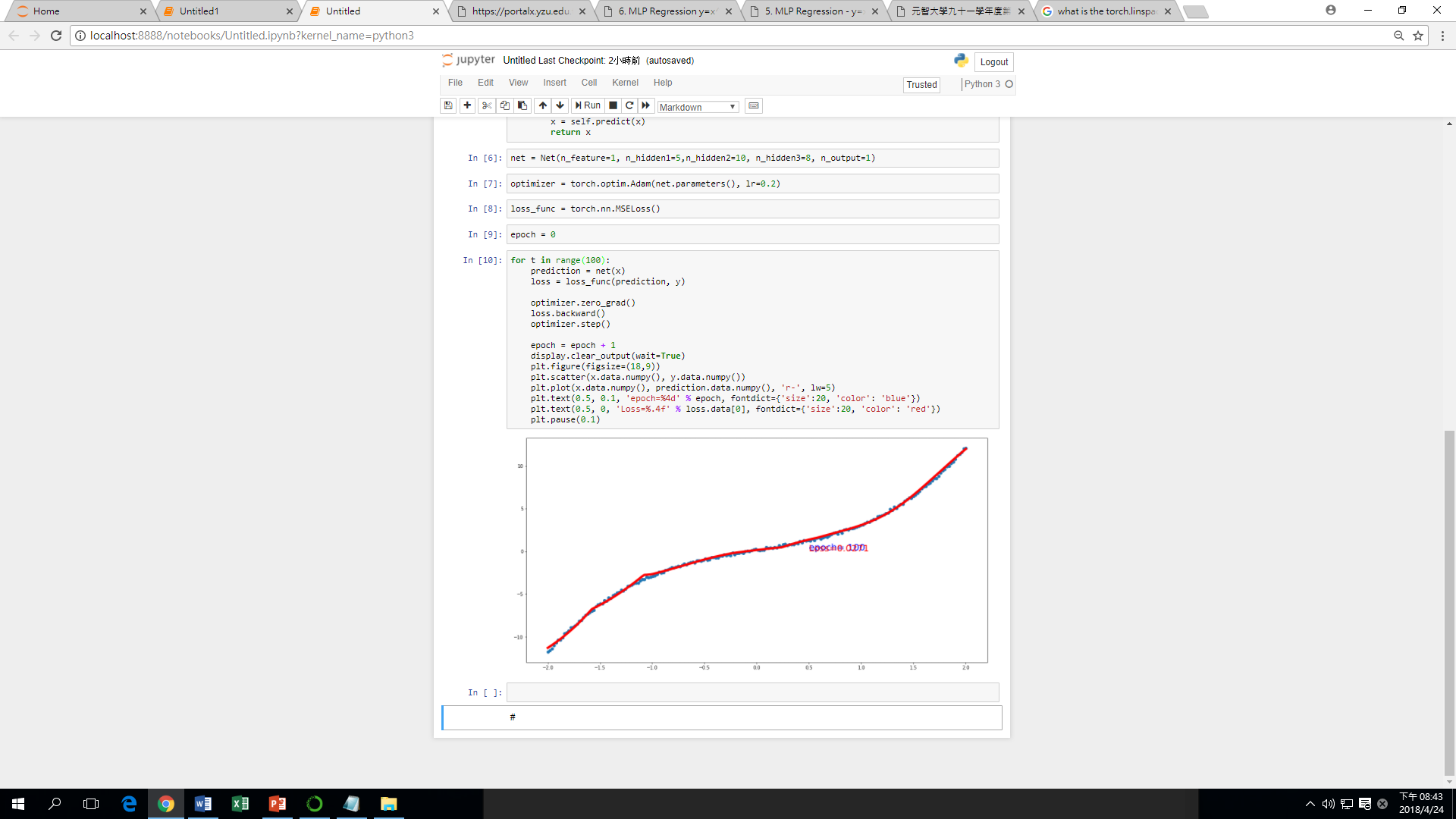
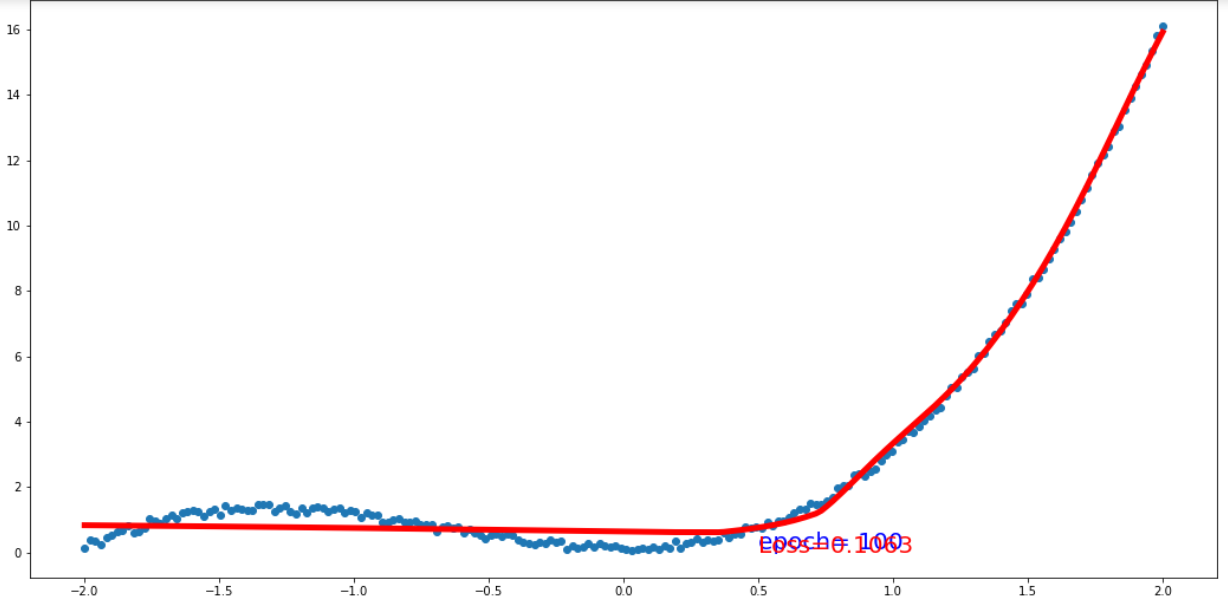


Figure 9. The Result of Altering the Parameter

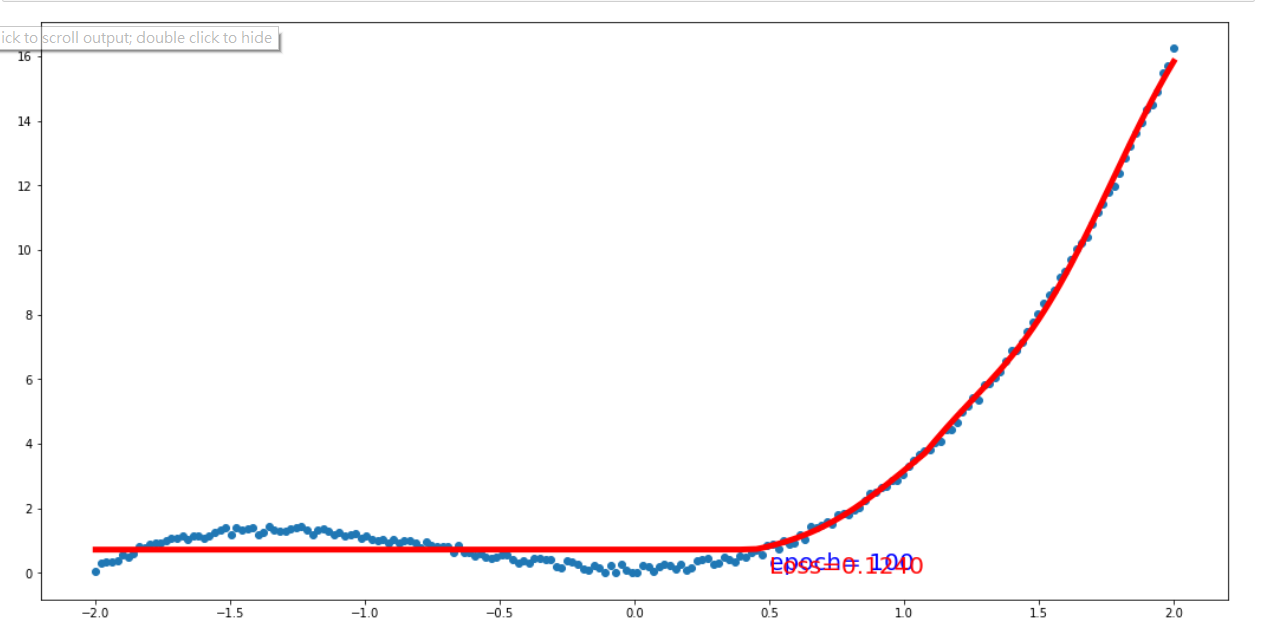
The result is shown in the Figure 8. After changing several parameters, the neural network can learn better and predict the y curve more accurately. The loss become 0.001 from the previous value of 22.3.

# 王偉亮 (Patrick Purnama)

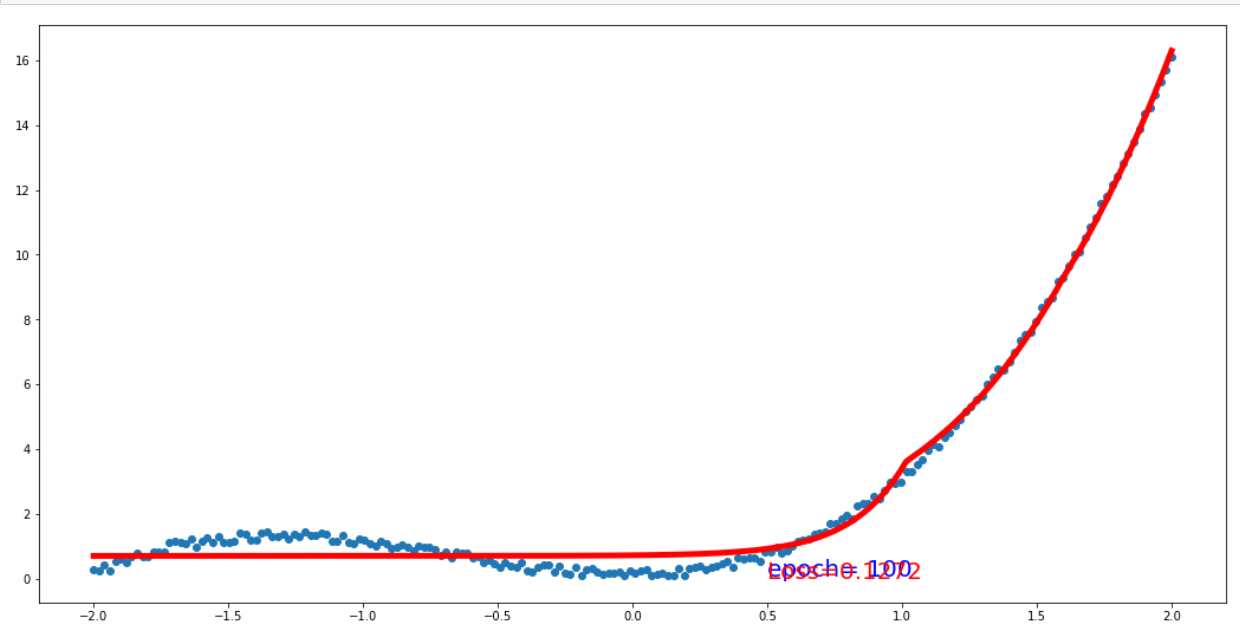
For the first trial, I try to change the parameter of my Neural Network into adam with 0.2 learning rate and epoch number 100, the result is:



The second trial, I try to change the learning rate to 0.3:



The third trial, I try to change the learning rate to 0.1:

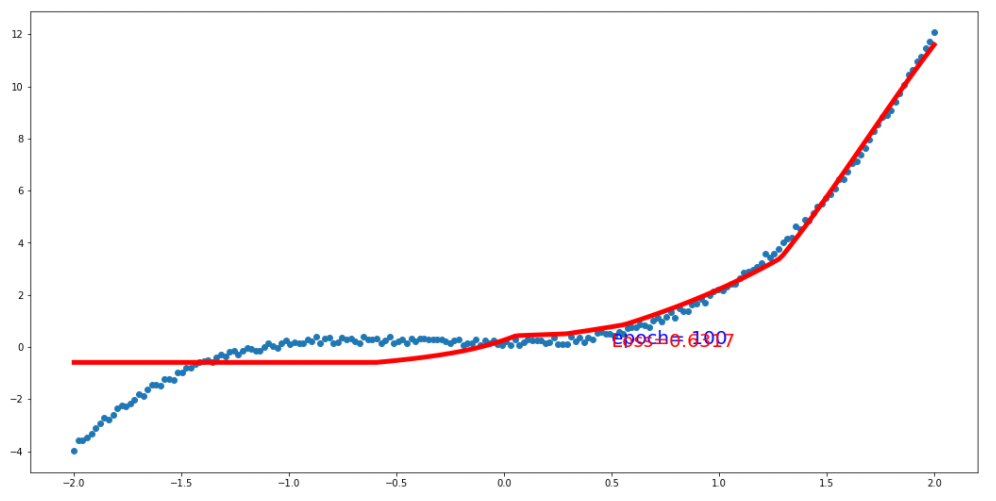


So, from all the results, the first trial is the best one with loss: 0.1063.

# 張純潔 (Ellice Jane Tiu)

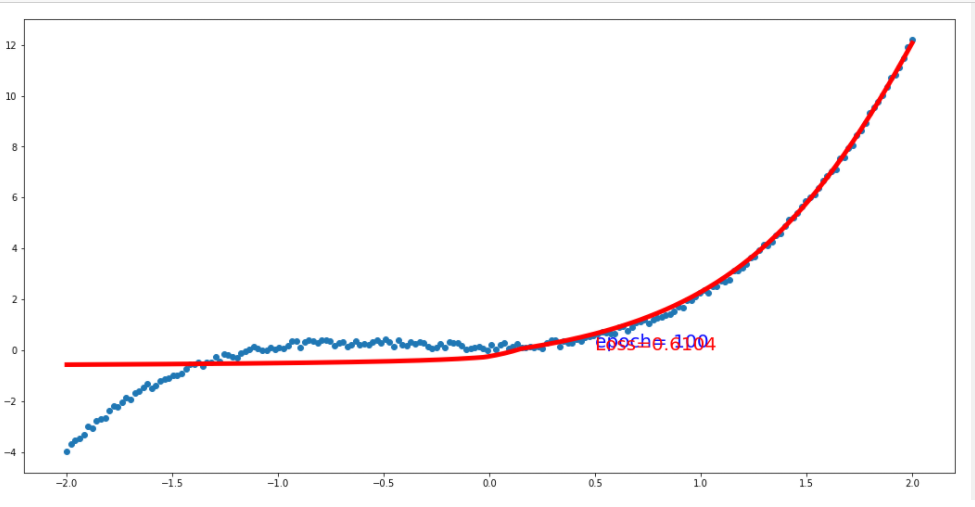
1. Change optimization function and learning rate

* SGD to Adam
* Learning rate from 1 to 0.1
* Epoch 30 to 100
* Output loss – 0.637

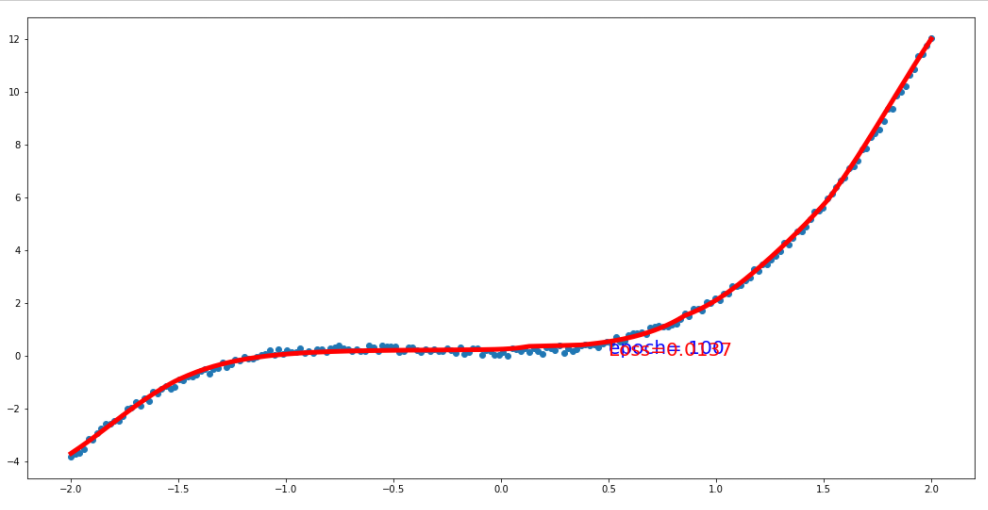


1. Change optimization function and learning rate

* SGD to Adam
* Learning rate from 0.1 to 0.2
* Epoch 30 to 100
* Output loss – 0.637

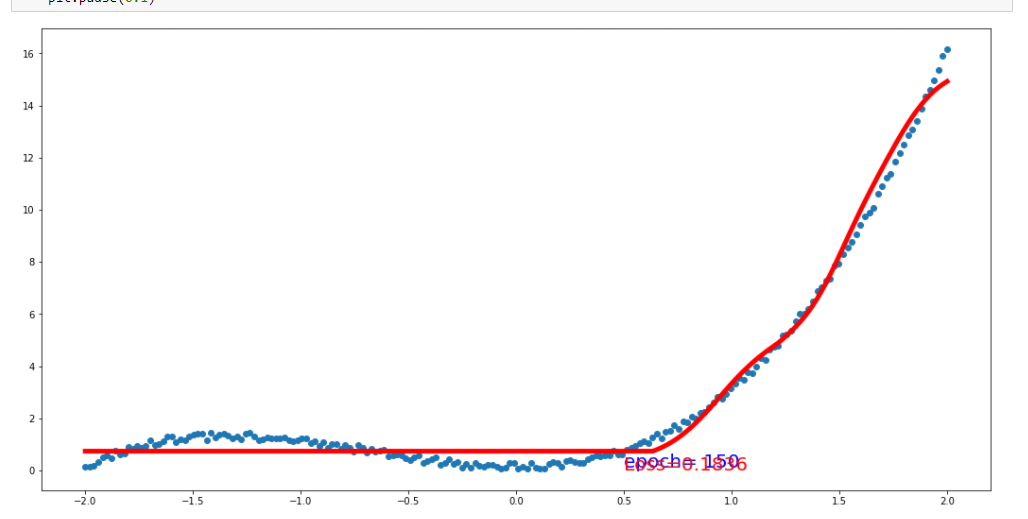


1. Change optimization function and learning rate

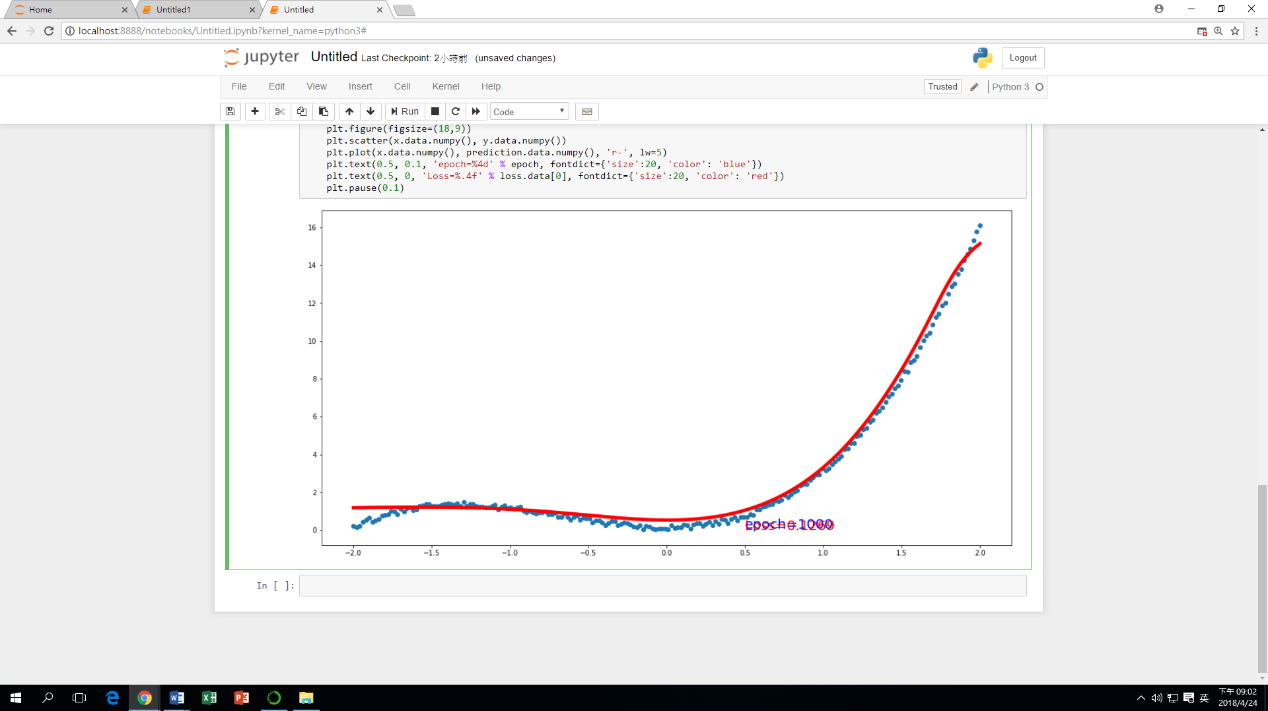
* SGD to Adam
* Learning rate from 0.2 to 0.3
* Epoch 30 to 100
* Output loss – 0.0137
* 

# 黃彥凱

我更改了最佳化的方法為Adam並將NN的數量調整為5.8.4，在一開始有嘗試將層數調多，但我發現當隱藏層的數量超過5層時將完全沒有學習效果，我也嘗試過將NN數量調大，但是成效也不佳，最佳化也試過Adam,RMS與SGD做比較，loss function也有過許多嘗試但最終還是使用MSE表現較優，並將epoch調大為150，可以有更好的學習效果。



# 謝忠宏



1. 跳動過大，試learnrat=0.1、0.05、0.01，0.01跳動幅度較小，採用0.01
2. 激活函數，首層試過sigmoid跟tanh，tanh fit有後半段比較好的趨勢，全都改採tanh
3. 增加EPOCH由30增加至1000，增加神經元W1(10個)、W2(10個)、W3(10

個)