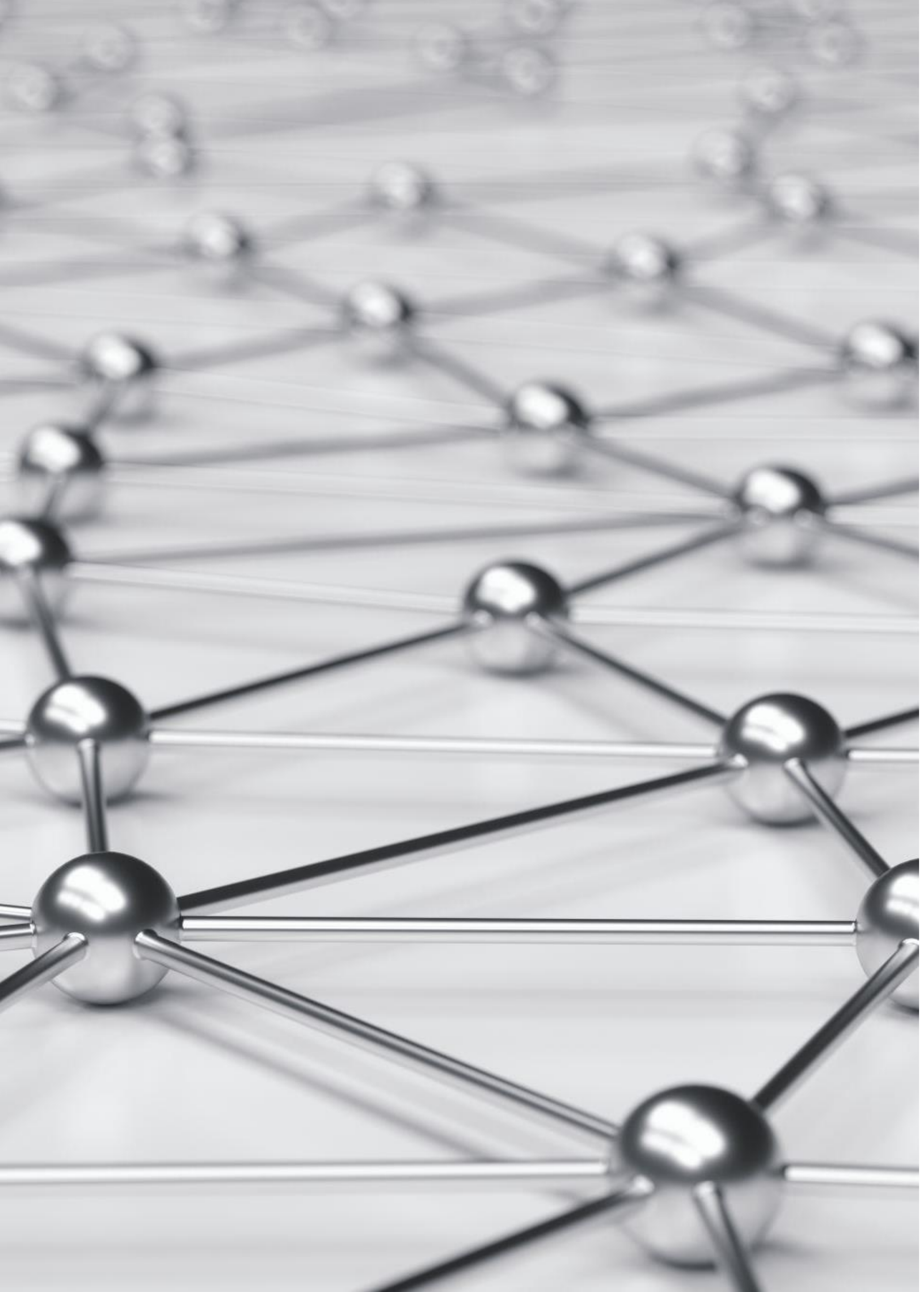
ML5: NEURAL NETWORK

Manuel, Ma. Janelle G.

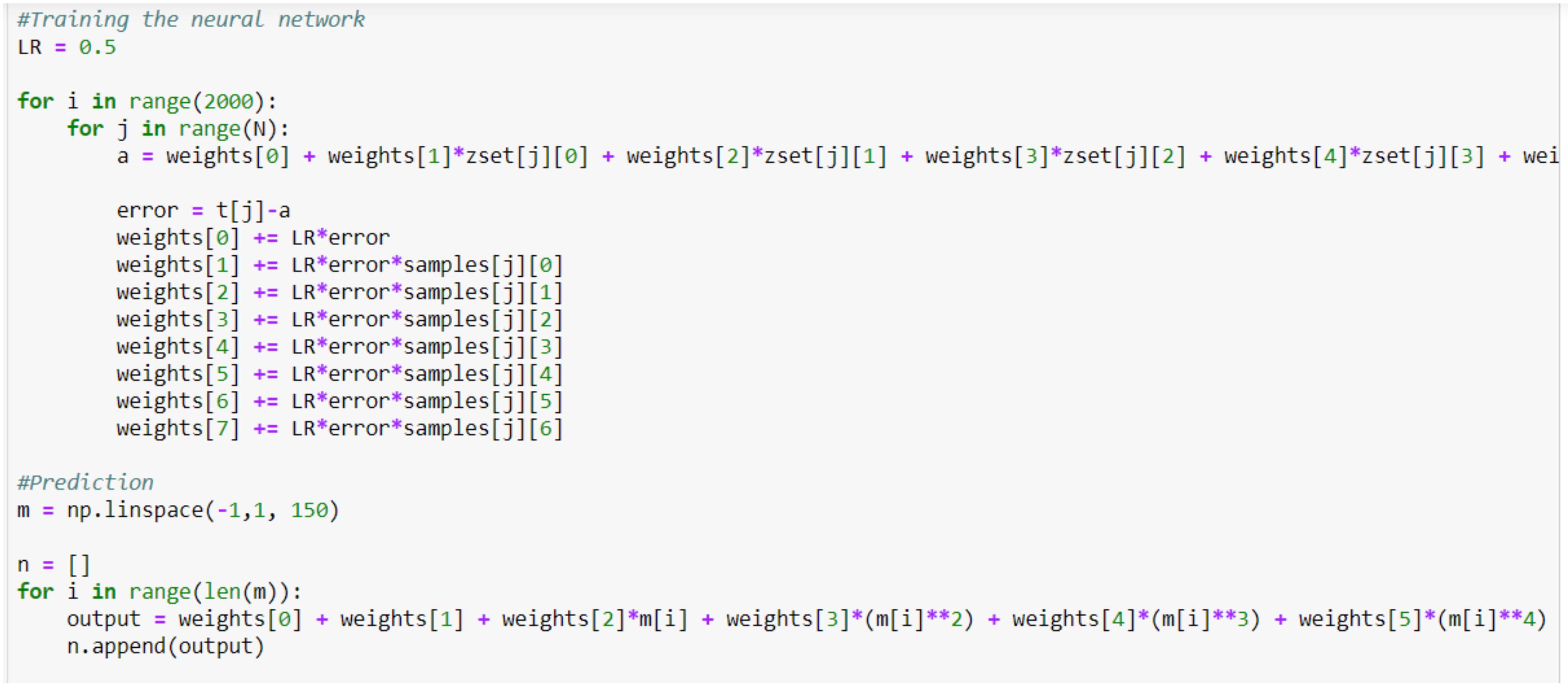


OBJECTIVES

The main objectives of this activity is to train a neural network to learn a sine function and to classify fruits using error backpropagation.

APPROXIMATING A SINE FUNCTION

NEURAL NETWORK REGRESSION

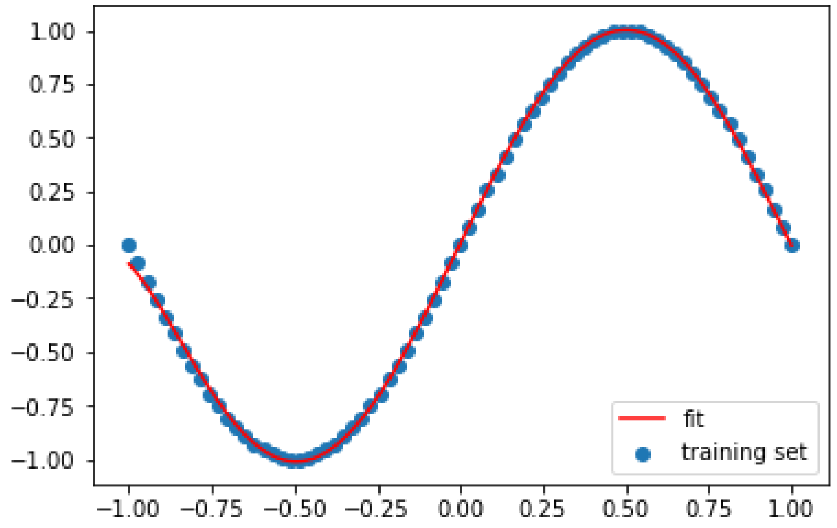


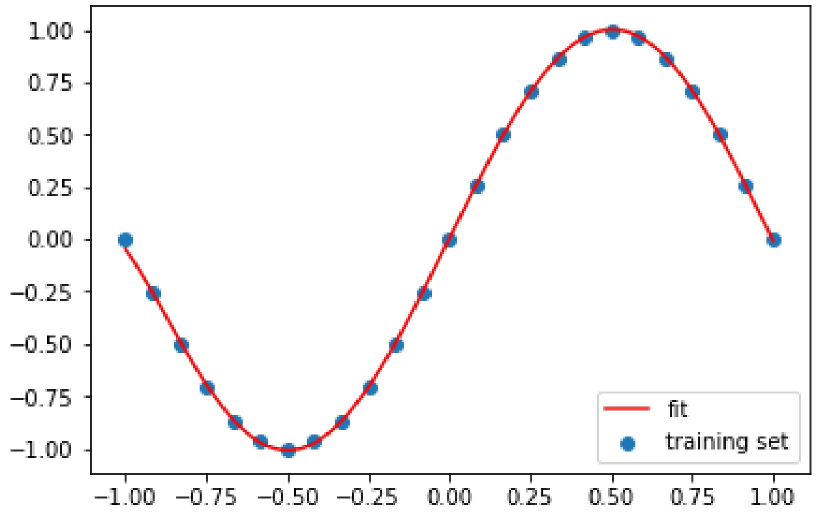
```
#Training the neural network
LR = 0.5

for i in range(2000):
    for j in range(N):
        a = weights[0] + weights[1]*zset[j][0] + weights[2]*zset[j][1] + weights[3]*zset[j][2] + weights[4]*zset[j][3] + wei

        error = t[j]-a
        weights[0] += LR*error
        weights[1] += LR*error*samples[j][0]
        weights[2] += LR*error*samples[j][1]
        weights[3] += LR*error*samples[j][2]
        weights[4] += LR*error*samples[j][3]
        weights[5] += LR*error*samples[j][4]
        weights[6] += LR*error*samples[j][5]
        weights[7] += LR*error*samples[j][6]

#Prediction
m = np.linspace(-1,1, 150)

n = []
for i in range(len(m)):
    output = weights[0] + weights[1] + weights[2]*m[i] + weights[3]*(m[i]**2) + weights[4]*(m[i]**3) + weights[5]*(m[i]**4)
    n.append(output)
```

Figure 1. Python algorithm for training the single-layer neural network to learn any given function.

Any activation function can be used for this neural network. In this case, I used a linear function.

Epoch was set to be 2000 while learning rate is 0.5.





RESULTS: APPROXIMATION OF SINE

The sine function was successfully approximated using a single-layer neural network (perceptron).

At 25 data points, the program was still able to accurately fit a sine function.

Figure 2. Sine approximation using single-layer neural network
In Python. N (training data points)= 75 (up). N=25 (down).

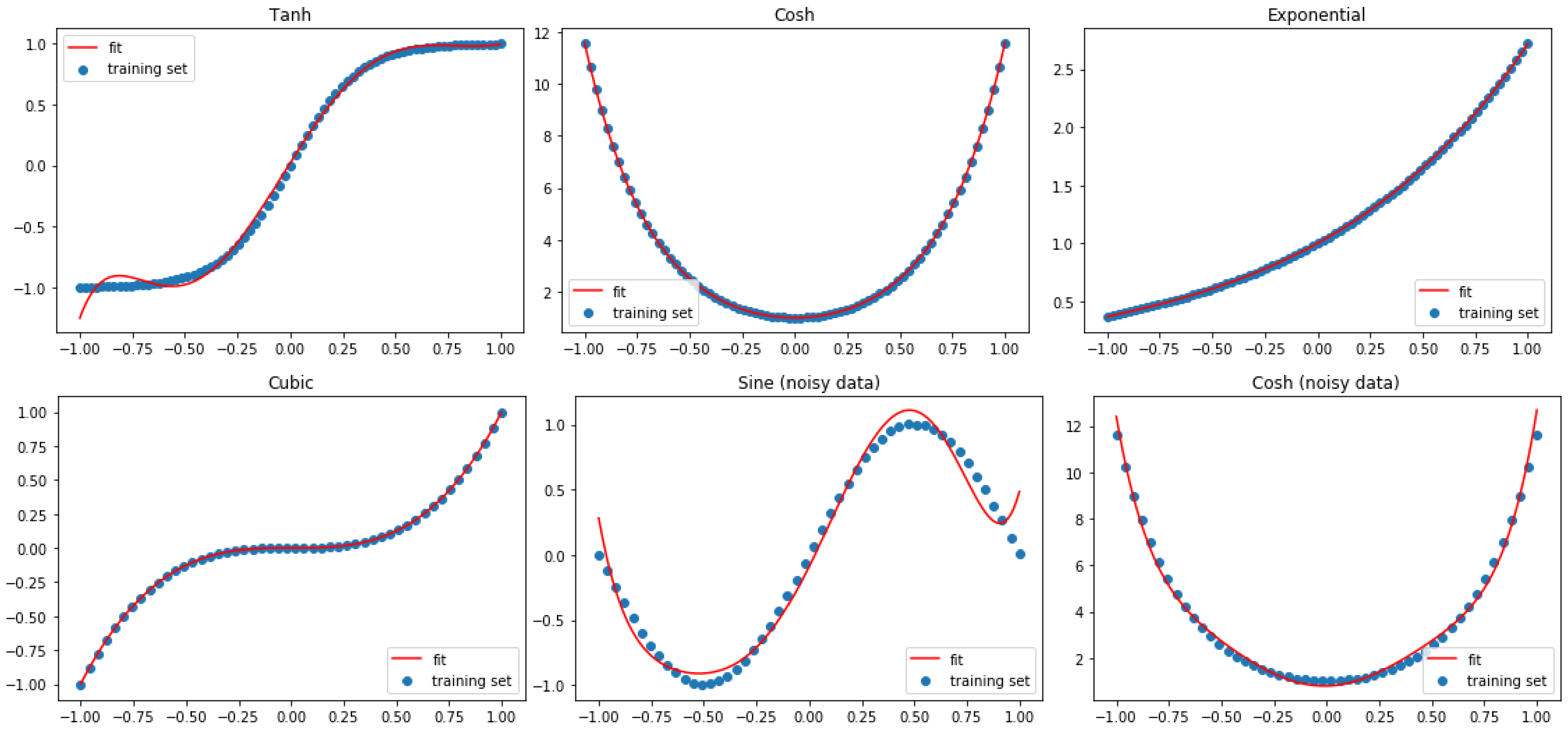


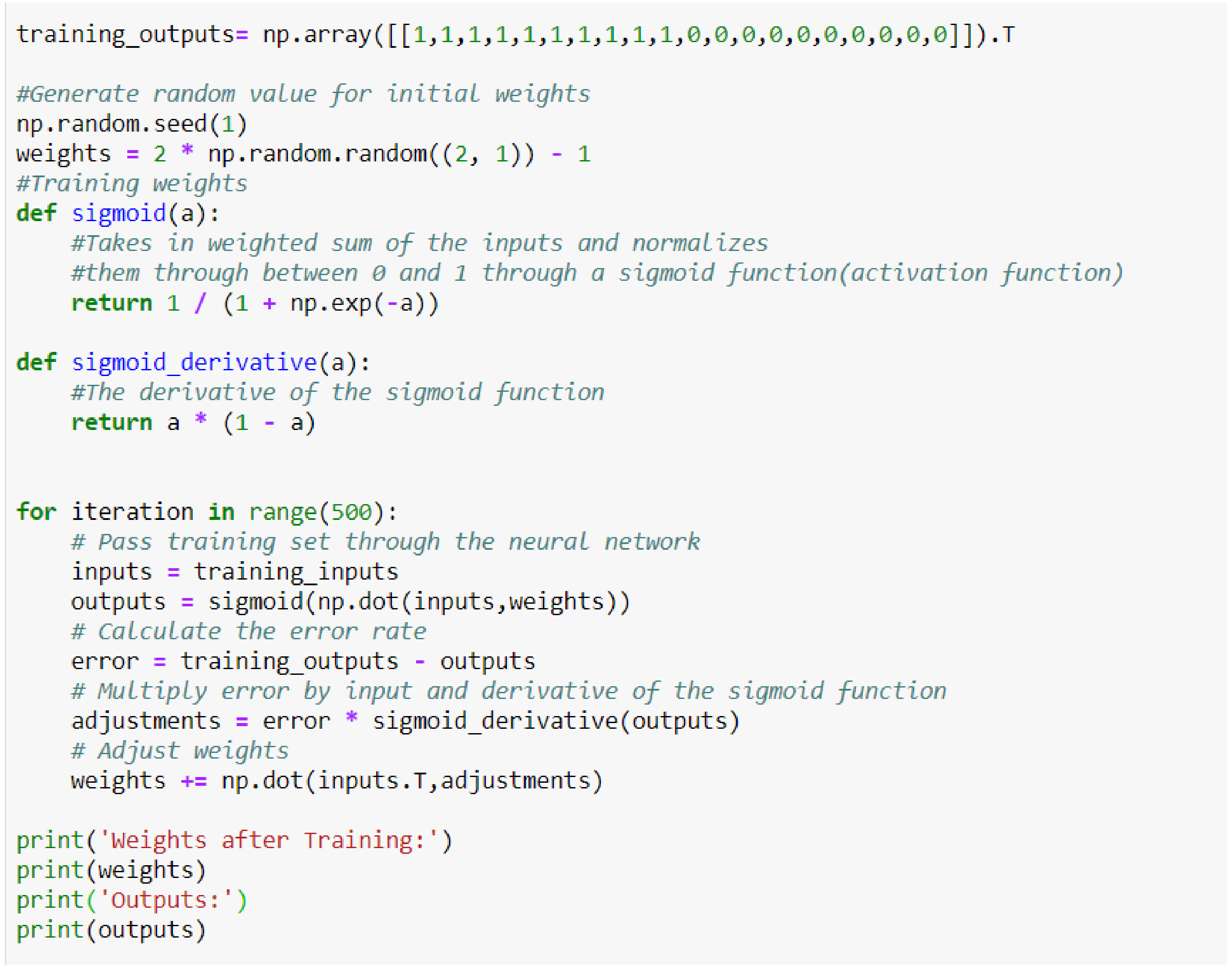
Figure 3. Other function approximations using the same single-layer neural network algorithm

CLASSIFYING FRUITS

TRAINING THE NEURAL NETWORK

Bananas and oranges were assigned to have a training output equal to 1 while mangoes were assigned to 0.

The activation function used in this algorithm is a sigmoid function.



```
training_outputs= np.array([[1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0]]).T

#Generate random value for initial weights
np.random.seed(1)
weights = 2 * np.random.random((2, 1)) - 1
#Training weights
def sigmoid(a):
    #Takes in weighted sum of the inputs and normalizes
    #them through between 0 and 1 through a sigmoid function(activation function)
    return 1 / (1 + np.exp(-a))

def sigmoid_derivative(a):
    #The derivative of the sigmoid function
    return a * (1 - a)


for iteration in range(500):
    # Pass training set through the neural network
    inputs = training_inputs
    outputs = sigmoid(np.dot(inputs,weights))
    # Calculate the error rate
    error = training_outputs - outputs
    # Multiply error by input and derivative of the sigmoid function
    adjustments = error * sigmoid_derivative(outputs)
    # Adjust weights
    weights += np.dot(inputs.T,adjustments)

print('Weights after Training:')
print(weights)
print('Outputs:')
print(outputs)
```

Figure 4. Python algorithm for training the neural network to classify fruits

RESULTS: TRAINING THE NEURAL NETWORK

Shown in Figure 5 is the result of running the algorithm in Figure 1 using the banana and mango data sets. The fit is close to the assigned training outputs!

The neural network was successfully trained to classify a banana from a mango.

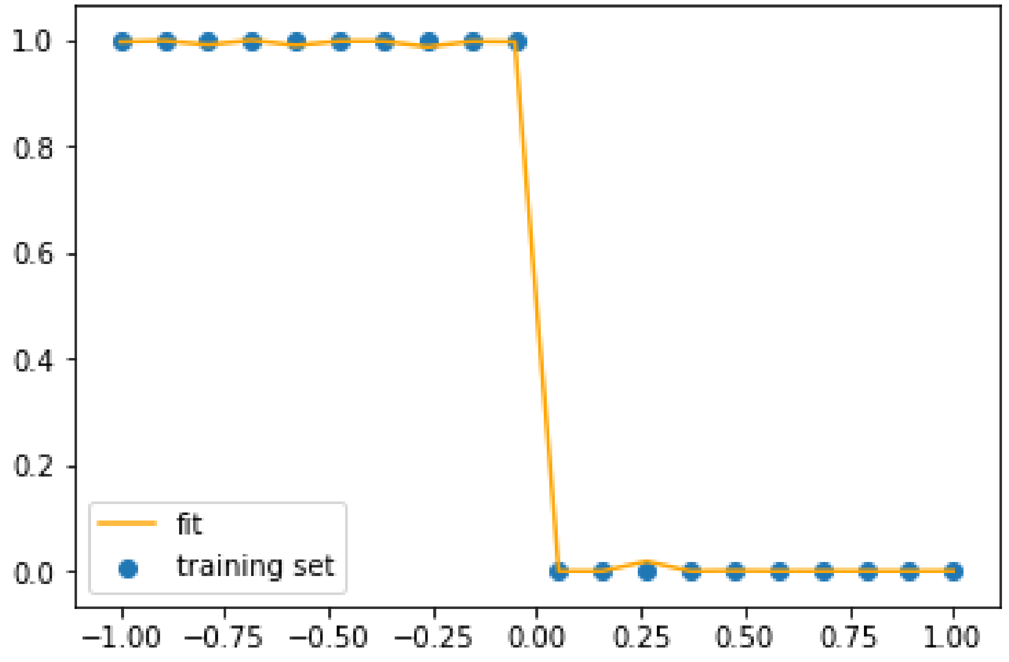


Figure 5. Output after training the neural network using banana and mango data set

RESULTS: CLASSIFYING FRUITS (ORANGES VS MANGOES)

Shown in Figure 6 is the result of running the algorithm in Figure 1 using the orange and mango data sets. The fit is close to the assigned training outputs!

The neural network was successfully trained to classify an orange from a mango.

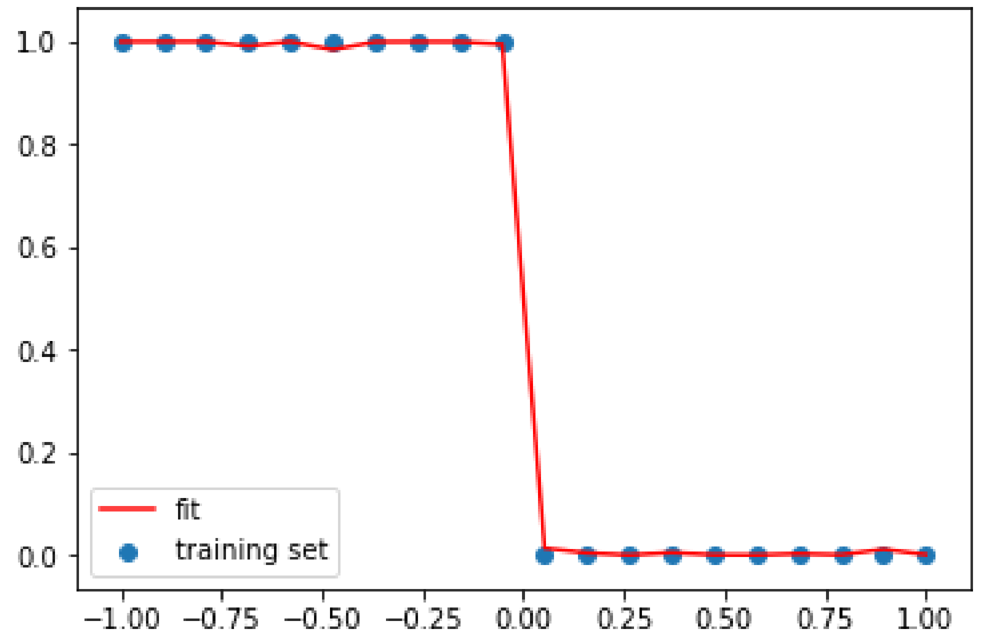


Figure 6. Output after training the neural network using orange and mango data set

TESTING THE ACCURACY OF THE NETWORK

H and Aspect ratio values are user-inputs.

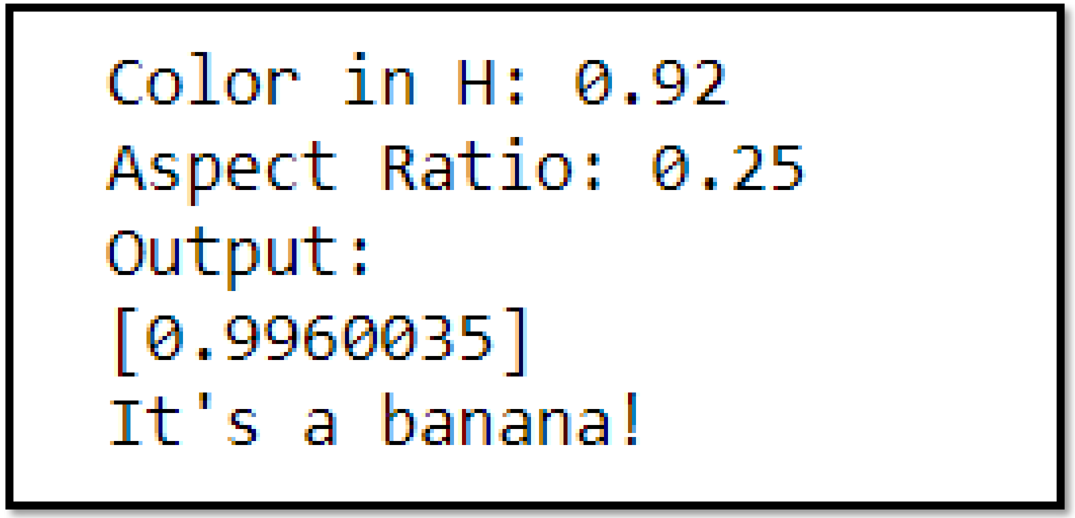
Shown in Figure 7 is the sample result in Python upon testing the neural network.

The accuracy are as follows:

Banana vs mangoes - 100%

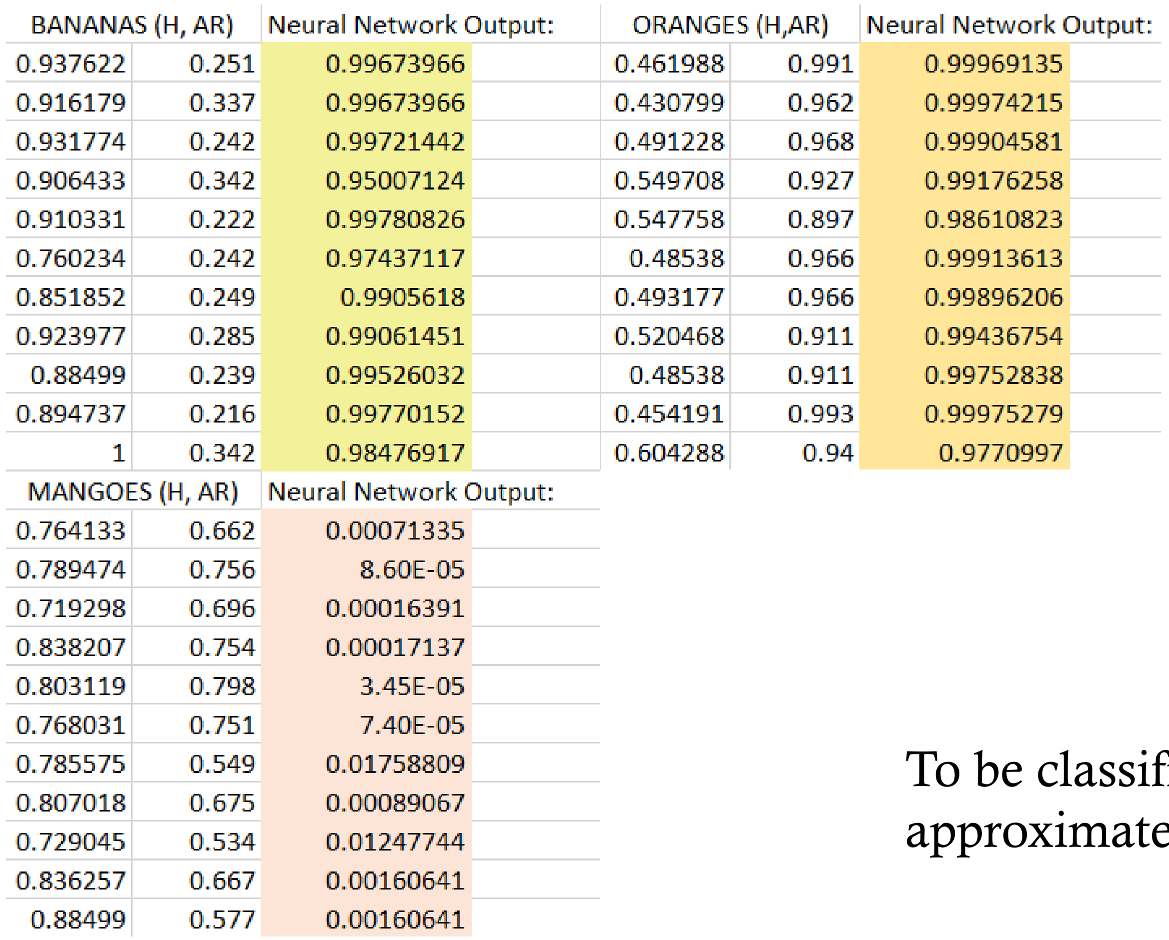
Oranges vs mangoes - 100%

*Detailed values are found in the next slide



```
Color in H: 0.92
Aspect Ratio: 0.25
Output:
[0.9960035]
It's a banana!
```

Figure 7. Sample result in Python upon testing the neural network.



BANANAS (H, AR)		Neural Network Output:	ORANGES (H,AR)		Neural Network Output:
0.937622	0.251	0.99673966	0.461988	0.991	0.99969135
0.916179	0.337	0.99673966	0.430799	0.962	0.99974215
0.931774	0.242	0.99721442	0.491228	0.968	0.99904581
0.906433	0.342	0.95007124	0.549708	0.927	0.99176258
0.910331	0.222	0.99780826	0.547758	0.897	0.98610823
0.760234	0.242	0.97437117	0.48538	0.966	0.99913613
0.851852	0.249	0.9905618	0.493177	0.966	0.99896206
0.923977	0.285	0.99061451	0.520468	0.911	0.99436754
0.88499	0.239	0.99526032	0.48538	0.911	0.99752838
0.894737	0.216	0.99770152	0.454191	0.993	0.99975279
1	0.342	0.98476917	0.604288	0.94	0.9770997

MANGOES (H, AR)		Neural Network Output:
0.764133	0.662	0.00071335
0.789474	0.756	8.60E-05
0.719298	0.696	0.00016391
0.838207	0.754	0.00017137
0.803119	0.798	3.45E-05
0.768031	0.751	7.40E-05
0.785575	0.549	0.01758809
0.807018	0.675	0.00089067
0.729045	0.534	0.01247744
0.836257	0.667	0.00160641
0.88499	0.577	0.00160641

Table 1. Tabulated results for the test set.

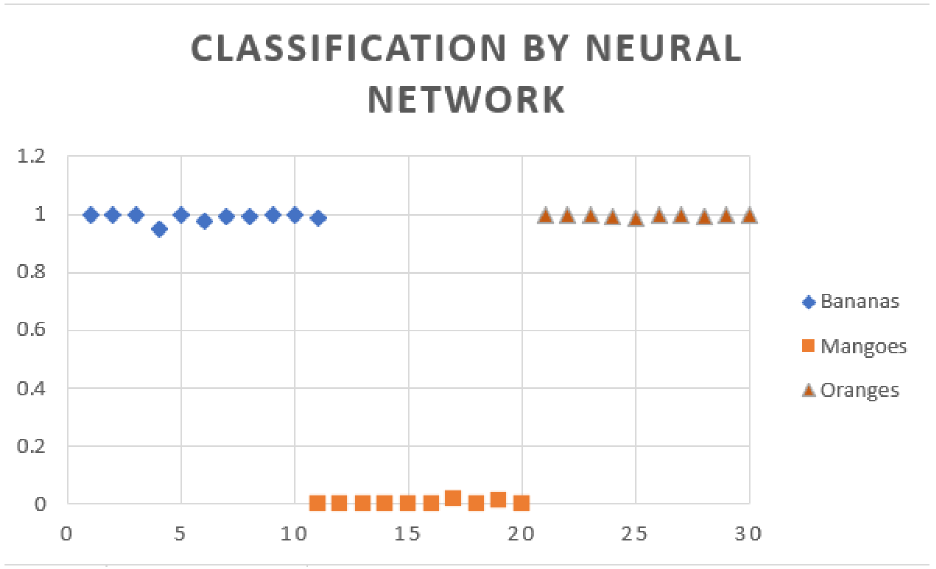


Figure 8. Fruit Classification of the Neural Network

To be classified as a banana or an orange, output should be approximately equal to 1 while mangoes to 0.

As shown in Figure 8, the programmed Neural Network produced accurate results!

ANALYSIS

Neural networks for regression and classification were successfully programmed and executed.

Although the accuracy for the classification is 100%, it would be best if the neural network can classify all three fruits at once. A little bit of tweaking in the algorithm is needed to achieve this.

For the function approximation, I was curious if the algorithm can still produce an accurate fit if the test data is noisy. As seen in the last two figures in Figure 3, the fit for sine and hyperbolic cosine functions are still pretty great!

REFLECTION

Rating: 12/10

This activity is a difficult one. It's quite easy to check if the algorithm works or not since you can compare the actual with the predicted outputs. I am really grateful for all the readily available references and videos for neural networks especially this one: https://www.youtube.com/watch?v=kft1AJ9WVDk! When I got lost with all the loops and functions, I was saved by this video!

REFERENCES

- Soriano, M. (2020). ML5 – Neural Networks
- Bishop, C. (1994). Neural networks and their applications
- Chris. (2019). Can neural networks approximate mathematical functions? – MachineCurve