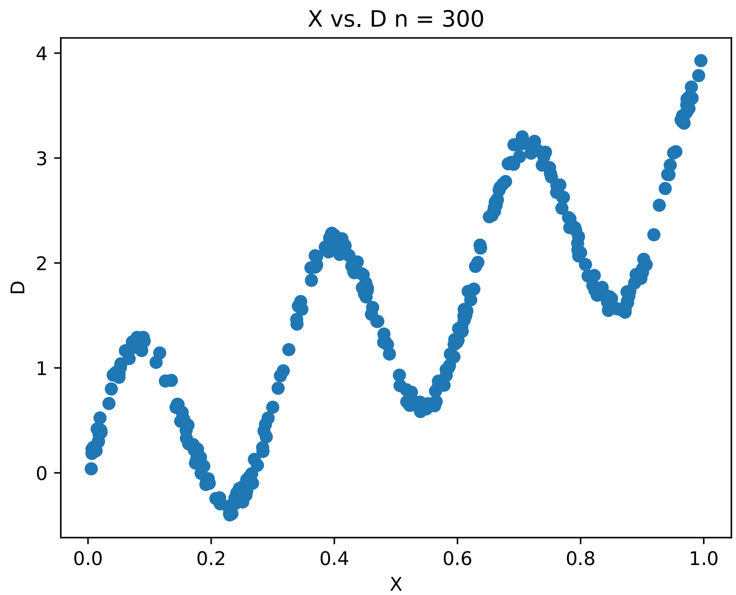
CS 559 Homework 4

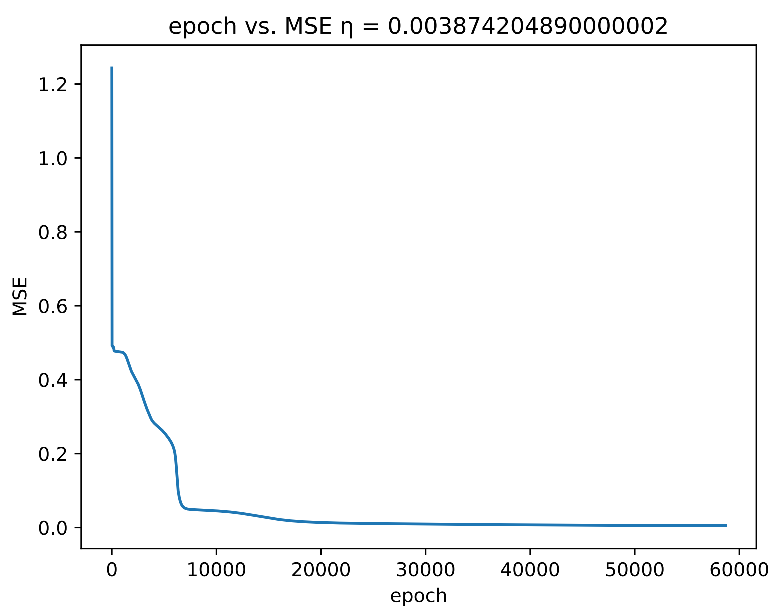
Huiyang Zhao
655490960

Q1.

3. Let di = sin(20xi) + 3xi + νi, i = 1, . . . , n. Plot the points (xi, di), i = 1, . . . , n.

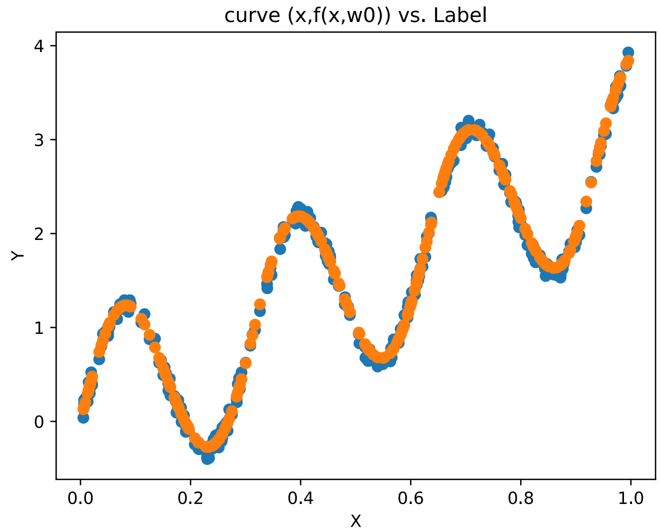


4. Plot the number of epochs vs the MSE in the backpropagation algorithm.



The MSE finally become less than 0.005 after less than 60000 epochs of training.
Learning rate η decreases from 0.01 to 0.003874… as shown above.
The w_0 we got in this process leads to the curve below, which is a good fit.

5. Plot the curve f(x, w0) as x ranges from 0 to 1 on top of the plot of points in (c).



Label is colored blue while curve is colored orange.

6. Your report should include a pseudocode of your training algorithm including all gradient descent update equations written out explicitly.

PseudoCode:

Initialize n, X, V, η, D = sin(20X) + 3X + V
Initialize N, eps, FLAG = True
Initialize weight W_{3N+1*1} = [w1, b1, w2, b2]^T
While FLAG:
    error = 0
    for xi in X:
        yi = $\phi_2(w_2\ \phi_1(w_1 x_i + b_1)\ + b_2)$
        $= w_2\ tanh(w_1 x_i + b_1)\ + b_2$
        error = error + (D[i] - yi)^2
        for w in W:
            calculate $\frac{\partial E}{\partial w} = -$ (the signal before multiplication by w in the feedforward network) × (the signal before multiplication by w in the feedback network).
        update w1, b1, w2, b2 with calculated $\frac{\partial E}{\partial w}$:
            w <- w – η * $\frac{\partial E}{\partial w}$ (Detailed **gradient descent update equations** are attached)
    end for
    MSE = error/n
    If MSE>previous MSE:
        η = η * 0.9
    If MSE< eps
        FLAG = False
end while

Gradient descent update equations:

$b_2 = b_2 + \eta \bullet 1 \bullet (D[i] - y_i) \bullet 1$

$w_2 = w_2 + \eta \bullet \tanh(w_1 x_1 + b_1) \bullet (D[i] - y_i) \bullet 1$

$b_1 = b_1 + \eta \bullet 1 \bullet w_2 \bullet (D[i] - y_i) \bullet 1 \bullet 1\text{-}\tanh^2(w_1 xi + b_1)$

$w_1 = w_1 + \eta \bullet x_i \bullet w_2 \bullet (D[i] - y_i) \bullet 1 \bullet 1\text{-}\tanh^2(w_1 xi + b_1)$

```python
# CS559 Neural Network
# Huiyang Zhao
# UIN 655490960


import numpy as np
import matplotlib.pyplot as plt


def activation_1(v):
    return np.tanh(v)


def activation_1_prime(v):
    return 1 - pow(np.tanh(v), 2)


def activation_2(v):
    return v


seed = 655490960
np.random.seed(seed)
n = 300
# 1. Draw n=300 real numbers uniformly at random on [0, 1], call them x1,...,xn.
X = np.random.uniform(0, 1, (n, 1))
# 2. Draw n real numbers uniformly at random on [− 1 , 1 ], call them ν1,...,νn.
V = np.random.uniform(-1 / 10, 1 / 10, (n, 1))
# 3. Let di = sin(20xi) + 3xi + νi, i = 1, ..., n.
D = np.sin(20 * X) + 3 * X + V

# Plot the points(xi, di), i = 1, ..., n.
plt.figure()
plt.title('X vs. D n = {}'.format(n))
plt.xlabel('X')
plt.ylabel('D')
plt.scatter(X, D)
plt.savefig('X_vs_Label.pdf')
plt.show()

lr = 0.01
eps = 0.005

if __name__ == '__main__':
    N = 24
    # Let w denote the vector of all these 3N + 1 weights.
    W = np.random.uniform(-0.1, 0.1, 3 * N + 1)
    # weights before first layer
    w1_init = W[0:N]
    # weights before second layer
    w2_init = W[2 * N:3 * N]
    # bias for first layer
    b1_init = W[N:2 * N]
    # bias for second layer
    b2_init = W[3 * N]

    epoch = 0
    error_array = []

    w1 = w1_init
    w2 = w2_init
    b1 = b1_init
    b2 = b2_init
```

```python
flag = True
print('------Training------')
while flag:
    error = 0
    # online training
    for i in range(n):
        # feed forward
        v1 = w1 * X[i] + b1
        y1 = activation_1(v1)
        v2 = w2.transpose() * y1
        y2 = activation_2(np.sum(v2) + b2)
        # calculate errors
        error += pow(D[i] - y2, 2)
        # back forward
        b2 = b2 + lr * 1 * (D[i] - y2) * 1
        w2 = w2 + lr * y1 * (D[i] - y2) * 1
        b1 = b1 + lr * 1 * w2 * (D[i] - y2) * 1 * activation_1_prime(v1)
        w1 = w1 + lr * X[i] * w2 * (D[i] - y2) * 1 * activation_1_prime(v1)

    MSE = error / n
    error_array.append(MSE)
    # modify the gradient descent algorithm by decreasing η
    if epoch > 1 and (error_array[-1] > error_array[-2]):
        lr = lr * 0.9
    epoch += 1
    if epoch > 1000000 or MSE < eps:
        flag = False

# Plot the number of epochs vs the MSE in the bp algorithm.
epoch_array = range(epoch)
plt.figure()
plt.title('epoch vs. MSE η = {}'.format(lr))
plt.xlabel('epoch')
plt.ylabel('MSE')
plt.plot(epoch_array, error_array)
plt.savefig('Epoch_vs_MSE.pdf')
plt.show()

outputs = []
for i in range(n):
    v1 = w1 * X[i] + b1
    y1 = activation_1(v1)
    v2 = w2.transpose() * y1
    y2 = activation_2(np.sum(v2) + b2)
    outputs.append(y2)

# Plot the curve f(x, w0)
plt.figure()
plt.title('curve (x,f(x,w0)) vs. Label')
plt.xlabel('X')
plt.ylabel('Y')
plt.scatter(X, D, label='Label')
plt.scatter(X, outputs, label='Output')
plt.savefig('Output_vs_Label.pdf')
plt.show()
```