109062318 簡弘哲 HW7

1.

```
import random
from random import randrange
from math import *
import matplotlib.pyplot as plt

w_z1=[0,0]
w_z2=[0,0]

#initialize weight coefficients
for i in range(2):
    w_z1[i]=random.uniform(-0.1,0.1)
    w_z2[i]=random.uniform(-0.1,0.1)
v_1=random.uniform(-0.1,0.1)
v_2=random.uniform(-0.1,0.1)
v_0=random.uniform(-0.1,0.1) # adding bias term for v

eta=0.05 #define learning rate
```

```
# set the rest of element to be zero
x = [ [1,0.0], [1,0.0], [1,0.0], \
      [1,0.0], [1,0.0], [1,0.0], \
      [1,0.0], [1,0.0], [1,0.0],
      [1,0.0], [1,0.0], [1,0.0], \
      [1,0.0], [1,0.0], [1,0.0], \
      [1,0.0], [1,0.0], [1,0.0],
      [1,0.0], [1,0.0], [1,0.0], \
      [1,0.0], [1,0.0], [1,0.0], \
      [1,0.0], [1,0.0], [1,0.0] ]

# desired output array
r= [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0, \
    0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0, \
    0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]

for i in range(27):
    x[i][1]=random.uniform(-0.5,0.5)
    r[i]=sin(6*x[i][1])+random.gauss(0,0.1)
```

```
for i in range(27000):
    j=randrange(27) #randomly pick sample vector of out of 8
    desiredoutput=r[j]
    sum_w_z1=0
    sum_w_z2=0
    sum_v=0
    for k in range(2):
        sum_w_z1 = sum_w_z1 + w_z1[k]*x[j][k]
        sum_w_z2 = sum_w_z2 + w_z2[k]*x[j][k]
    z1_h=tanh(sum_w_z1)
    z2_h=tanh(sum_w_z2)
    sum_v=v_1*z1_h+v_2*z2_h+v_0 #keep only two hidden unit
    output_y=sum_v  #use linear unit as output


    # delta rule
    # update = learning rate*(Desired output - Actualouput)*input
    v_1=v_1-eta*(output_y-desiredoutput)*z1_h
    v_2=v_2-eta*(output_y-desiredoutput)*z2_h
    v_0=v_0-eta*(output_y-desiredoutput)*1

    for m in range(2):
        # update = learning rate*v*(1-z**2)*(Desired output - Actualouput)*input
        w_z1[m] = w_z1[m] - eta*v_1*(1-z1_h*z1_h)*(output_y-desiredoutput)*x[j][m]
        w_z2[m] = w_z2[m] - eta*v_2*(1-z2_h*z2_h)*(output_y-desiredoutput)*x[j][m]
```

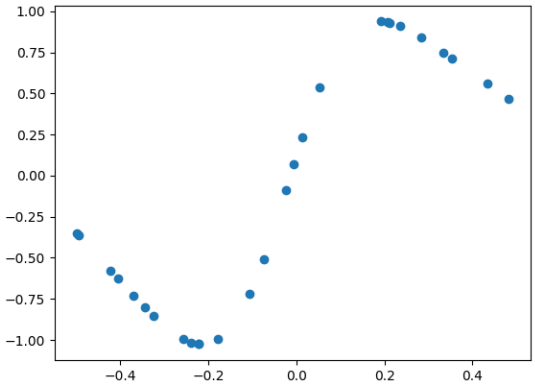
```
Y=[]
for j in range(27):
    desiredoutput = r[j]
    sum_w_z1=0
    sum_w_z2=0
    sum_v=0
    for k in range(2):
        sum_w_z1=sum_w_z1+ w_z1[k]*x[j][k]
        sum_w_z2=sum_w_z2+ w_z2[k]*x[j][k]
    z1_h=tanh(sum_w_z1)
    z2_h=tanh(sum_w_z2)

    sum_v=v_1*z1_h+v_2*z2_h+v_0
    output_y=sum_v
    Y.append(output_y)

#print(f'input x = {round(x[j][1],3)}, desiredoutput = {round(desiredoutput,3)}, actualoutput = {round(ou
# in total, this newtork has 7 coefficient, namely 3 v coefficents and 4 w cofficients
print(f'v0 = {round(v_0,3)}, v1 = {round(v_1,3)}, v2 = {round(v_2,3)}')
print(f'wz1_0_bias = {round(w_z1[0],3)}, wz1_1 = {round(w_z1[1],3)}\nwz2_0_bias = {round(w_z2[0],3)}, wz
```

```
wz2_0_bias = {round(w_z2[0],3)}, wz2_1 = {round(w_z2[1],3)}')
```

1(a).

```
v0 = 0.351, v1 = -2.63, v2 = -2.081
wz1_0_bias = 0.204, wz1_1 = 1.563
wz2_0_bias = -0.148, wz2_1 = -6.178
```

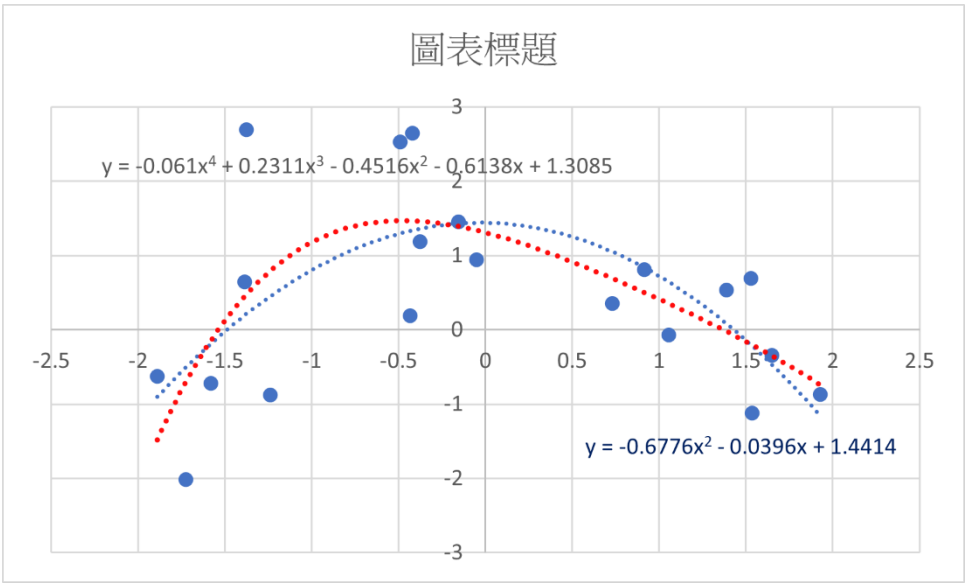
1(b).

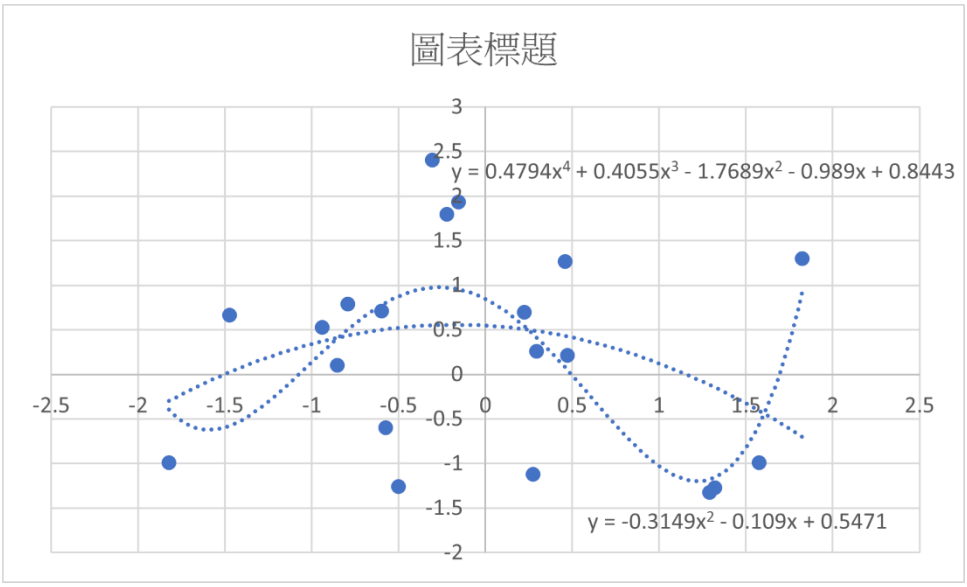


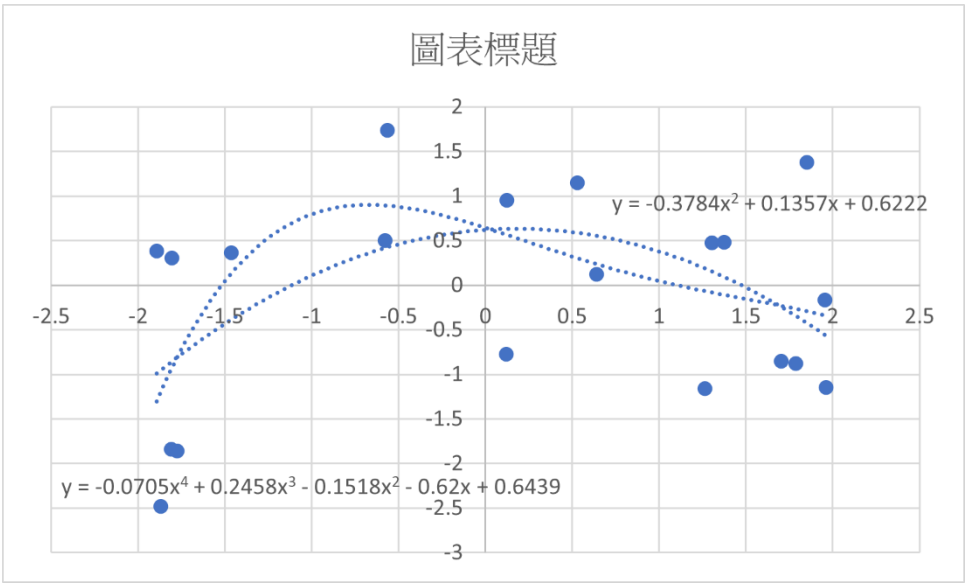
2.

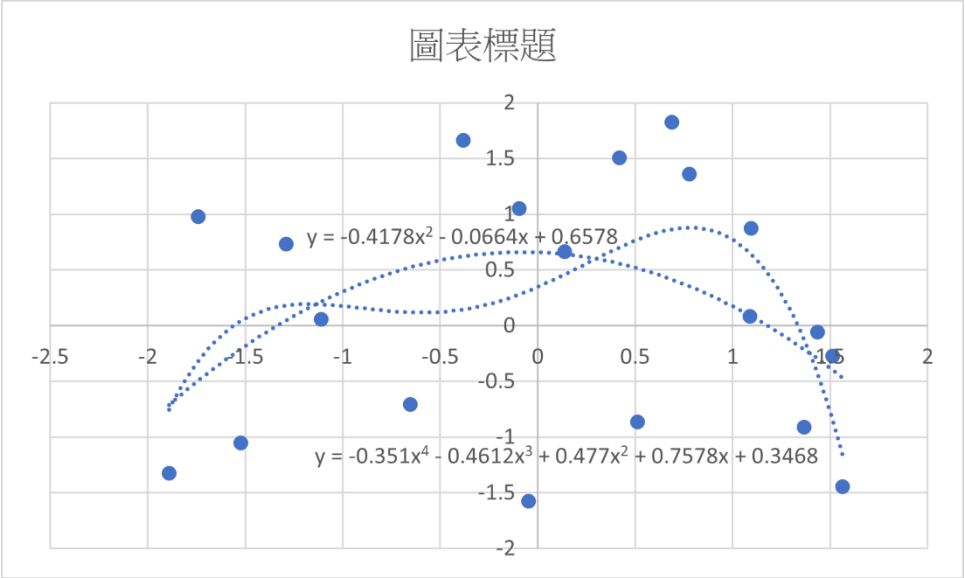
```python
import random
from math import *
import numpy as np

X, Y = [], []
for i in range(5):
    with open(f'dataset{i}.csv','w') as fh:
        fh.write('x,y\n')
        tmp_x, tmp_y = [], []
        for j in range(20):
            x=random.uniform(-2,2)
            y=cos(1.5*x)+random.gauss(0,1)
            fh.write(f'{x},{y}\n')
            tmp_x.append(x)
            tmp_y.append(y)
        X.append(tmp_x)
        Y.append(tmp_y)
```

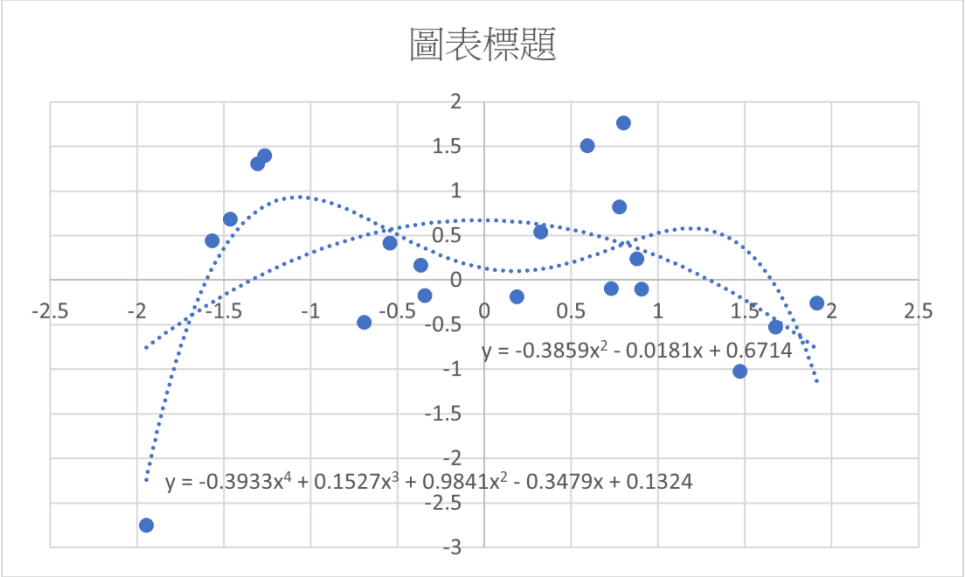
```python
def solve(deg):
    COEFS=[]
    for i in range(5):
        # coef: Polynomial coefficients, highest power first.
        # ex. deg(f(x))=2 and coef=[1,2,3] => f(x) = 1x**2 + 2x + 3
        coef=np.polyfit(x=X[i], y=Y[i], deg=deg)
        print(f"g{i+1}(x) coef = {coef}")
        COEFS.append(coef)
    print(f"deg={deg} => average gi(x) = {np.mean(COEFS,axis=0)}")

solve(deg=4)    # [x^4 coef, x^3 coef, x^2 coef, x coef, const]
print()
solve(deg=2)    # [x^2 coef, x coef, const]
```


圖表標題
y = -0.061x4 + 0.2311x3 - 0.4516x2 - 0.6138x + 1.3085
y = -0.6776x2 - 0.0396x + 1.4414


圖表標題
y = 0.4794x4 + 0.4055x3 - 1.7689x2 - 0.989x + 0.8443
y = -0.3149x2 - 0.109x + 0.5471


圖表標題
y = -0.3784x2 + 0.1357x + 0.6222
y = -0.0705x4 + 0.2458x3 - 0.1518x2 - 0.62x + 0.6439





2(a).

```
g1(x) coef = [-0.06104511  0.23109453 -0.45164203 -0.61379891  1.30847364]
g2(x) coef = [ 0.47938248  0.4055403  -1.76885149 -0.98902569  0.84426244]
g3(x) coef = [-0.07047247  0.24583116 -0.15182817 -0.62002046  0.64392355]
g4(x) coef = [-0.35100804 -0.46115451  0.47700791  0.75777052  0.34681844]
g5(x) coef = [-0.39332564  0.15267648  0.98410605 -0.34792618  0.1323921 ]
deg=4 => average gi(x) = [-0.07929376  0.11479759 -0.18224154 -0.36260015  0.65517403]
```

Average $g_i(x) = -0.079x^4 + 0.114x^3 - 0.182x^2 - 0.362x + 0.655$

2(b).

```
g1(x) coef = [-0.67763235 -0.03960987  1.44138044]
g2(x) coef = [-0.31485081 -0.10900803  0.54706929]
g3(x) coef = [-0.37837186  0.13574839  0.62215974]
g4(x) coef = [-0.41780943 -0.06640078  0.65780099]
g5(x) coef = [-0.38585685 -0.0181448   0.67143967]
deg=2 => average gi(x) = [-0.43490426 -0.01948302  0.78797003]
```

Average $g_i(x) = -0.434x^2 - 0.019x + 0.787$

3.

```python
import math
print("3(a)")
# P(|v-0.75|<0.06) = P(0.69<v<0.81) = P(v=0.7) + P(v=0.8)
# P(x) = C(N,x) * (p^x) * (1-p)^(N-x)
print(math.comb(10,7) * (0.75)**7 * (0.25)**3 + math.comb(10,8) * (0.75)**8 * (0.25)**2)

print("3(b)")
# P(v<1) = P(v=0) + P(v=0.1) + P(v=0.2) + ... + P(v=0.9) = 1 - P(v=1)
print(1 - math.comb(10,10) * (0.9)**10 * (0.1)**0)
```

```
3(a)
0.5318498611450195
3(b)
0.6513215599
```