108034058 鍾岷翰

1.

```
import random
from random import randrange
from math import *

def tanh(x):
        return exp(x)-exp(-x)/(exp(x)+exp(-x))
w_z1=[0,0]
w_z2=[0,0]
#w_z3=[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)
      # w_z3[i]=random.uniform(-0.1,0.1)
v_1=random.uniform(-0.1,0.1)
v_2=random.uniform(-0.1,0.1)
#v_3=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

# repurpose the input vector
# the first element is bias unit
# the second is input x
# target function f(x)= sin 6x from Alpaydin'book


# 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],
```

```
            ]

# 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):
        x1= random.uniform(-0.5,0.5)
        x[i][1]=x1
        y=sin(6*x1)+random.gauss(0,0.1)
        r[i]=y

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_w_z3=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]
                #sum_w_z3=sum_w_z3+ w_z3[k]*x[j][k]
        z1_h=tanh(sum_w_z1)
        z2_h=tanh(sum_w_z2)
        #z3_h=tanh(sum_w_z3)
        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
        # weight update Ethm Alpaydin' pseudo code
        # 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_3=v_3-eta*(output_y-desiredoutput)*z3_h
        v_0=v_0-eta*(output_y-desiredoutput)*1

        for m in range(2):
            # weight update Ethm Alpaydin' pseudo code
            # update= learning rate *v*z(1-z)*(Desired output - Actualouput)*input

            w_z1[m]=w_z1[m]-eta*v_1*(1-z1_h**2)*(output_y-desiredoutput)*x[j][m]
            w_z2[m]=w_z2[m]-eta*v_2*(1-z2_h**2)*(output_y-desiredoutput)*x[j][m]
            #w_z3[m]=w_z3[m]-eta*v_3*(1-z3_h**2)*(output_y-desiredoutput)*x[j][m]

    for j in range(27):
        desiredoutput = r[j]
        sum_w_z1=0
        sum_w_z2=0
        #sum_w_z3=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]
          #  sum_w_z3=sum_w_z3+ w_z3[k]*x[j][k]
        z1_h=tanh(sum_w_z1)
        z2_h=tanh(sum_w_z2)
        #z3_h=tanh(sum_w_z3)

        sum_v=v_1*z1_h+v_2*z2_h+v_0
        output_y= sum_v
```

```
        sum_v=v_1*z1_h+v_2*z2_h+v_0
        output_y= sum_v

        print('input x',round(x[j][1],3), 'desiredoutput', round(desiredoutput,3),'actualoutput', round(output_y,3))
# in total, this newtork has 7 coefficient, namely 3 v coefficents and 4 w cofficients
print('v0', round(v_0,3), 'v1', round(v_1,3), 'v2', round(v_2,3))
print('wz1_0_bias', round(w_z1[0],3), 'wz1_1', round(w_z1[1],3), 'wz2_0_bias', round(w_z2[0],3),'wz2_1', round(w_z2[1],3))
```

```
input x 0.329 desiredoutput 0.962 actualoutput 0.502
input x -0.134 desiredoutput -0.579 actualoutput 0.104
input x 0.12 desiredoutput 0.72 actualoutput 0.429
input x 0.04 desiredoutput 0.199 actualoutput 0.368
input x 0.0 desiredoutput -0.224 actualoutput 0.327
input x 0.12 desiredoutput 0.642 actualoutput 0.429
input x 0.062 desiredoutput 0.315 actualoutput 0.388
input x -0.475 desiredoutput -0.07 actualoutput -1.245
input x -0.297 desiredoutput -0.879 actualoutput -0.357
input x 0.448 desiredoutput 0.413 actualoutput 0.518
input x 0.407 desiredoutput 0.673 actualoutput 0.513
input x 0.356 desiredoutput 0.854 actualoutput 0.506
input x -0.003 desiredoutput -0.165 actualoutput 0.323
input x -0.369 desiredoutput -0.735 actualoutput -0.646
input x 0.176 desiredoutput 0.855 actualoutput 0.457
input x 0.344 desiredoutput 1.095 actualoutput 0.504
input x 0.133 desiredoutput 0.747 actualoutput 0.437
input x 0.257 desiredoutput 0.933 actualoutput 0.485
input x 0.086 desiredoutput 0.664 actualoutput 0.407
input x -0.189 desiredoutput -0.833 actualoutput -0.026
input x 0.149 desiredoutput 0.675 actualoutput 0.445
input x -0.416 desiredoutput -0.699 actualoutput -0.88
input x 0.197 desiredoutput 0.848 actualoutput 0.466
input x -0.469 desiredoutput -0.458 actualoutput -1.201
input x -0.097 desiredoutput -0.547 actualoutput 0.18
input x -0.309 desiredoutput -1.066 actualoutput -0.401
input x 0.171 desiredoutput 0.835 actualoutput 0.455
v0 -0.53 v1 -0.362 v2 -0.71
wz1_0_bias -0.849 wz1_1 -4.693 wz2_0_bias -5.132 wz2_1 -0.105
```

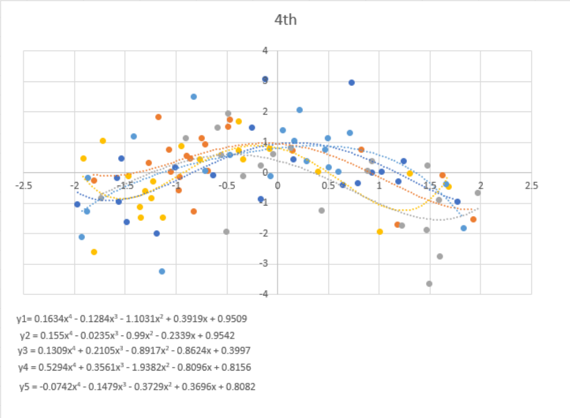
2.

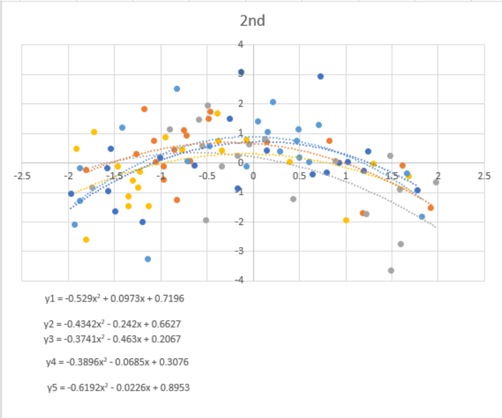
```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from math import cos

# Generate 5 data sets, each with 20 data points

df = pd.DataFrame(columns=['set', 'x', 'y'])
for i in range(5):
    for j in range(20):
        x = random.uniform(-2, 2)
        y = cos(1.5*x) + random.gauss(0, 1)
        df = df.append({"set": i+1 , "x":round(x, 4), "y":round(y, 4)},ignore_index=True)

df.to_csv('/content/drive/MyDrive/HW7_2.csv', index=False)
```





	w4	w3	w2	w1	w0	w2	w1	w0
average	0.18092	0.05336	-0.8592	-0.2289	0.78572	-0.4692	-0.1398	0.55838

左(a)右(b)

3.

3.

(a) $P[|v-\mu| < \epsilon] = P[\mu-\epsilon < v < \mu+\epsilon]$

$= P[10(0.75-0.06) < X < 10(0.75+0.06)]$

$= P[6.9 < X < 8.1]$

$= P[X \le 8] - P[X \le 6]$

$= P(X=8) + P(X=9) = C^{10}_{8}(0.75)^{8}(0.25)^{2} + C^{10}_{9}(0.75)^{9}(0.25)^{2}$

$= 0.53158$

(b) $P[v<1]$ $\mu = 0.9$ $P(X=0) = C^{10}_{0}(0.9)^{0}(0.1)^{10} = (0.1)^{10}$