```python
import numpy as np
import math
import matplotlib.pyplot as plt

np.random.seed(2)
x = np.random.uniform(size=(1,300))
x = x - np.mean(x)
v = np.random.uniform(low = -1/10, high = 1/10,size=(1,300))
d_true = np.sin(20*x) + 3*x + v

plt.scatter(x,d_true)
plt.show()
# 1 input
# N=24 hidden neurons and 1 output neuron
# 3N+1 weights
# actFun final = tanh v
step = 4*10**(-2)
N = 24
corr = 0
np.random.seed(3)
bias_hidden = np.random.uniform(size=(1,N))
wIn = np.random.uniform(size=(1,N))
wOut = np.random.uniform(size=(1,N))
bias_out = np.random.uniform()
miscl = 300
MSE=[100]
corr=0
epoch=0

while (MSE[-1]>0.02):
    epoch+=1
    errors=[]
    res=[]
    for el,ris in zip(x.T,d_true[0]):
        #forward
        a = (el * wIn) + bias_hidden
        d = np.tanh(a)
        d_w = d * wOut
        b = d_w.sum() + bias_out
        f_x = b
        res.append(f_x)

        #backpropagation
        diff = (ris - f_x)
        wOut = wOut + step * (2 * diff) * (d)
        wIn = wIn + step * (2 * diff) * wOut * (1 - (d**2)) * el
        bias_hidden = bias_hidden + step * (2 * diff) * wOut * (1 - (d**2))
        bias_out = bias_out + step * (2 * diff)

        errors.append(diff**2)
    MSE.append(sum(errors)/len(errors))

    if (MSE[-1]>MSE[-2]):
        step = step - 0.1*step

plt.scatter(x,d_true)
plt.scatter(x,res)
plt.legend(['true','learned'])
plt.show()
plt.plot([i for i in range(epoch)],MSE[1:])
plt.title('MSE vs number of epochs')
plt.show()
```