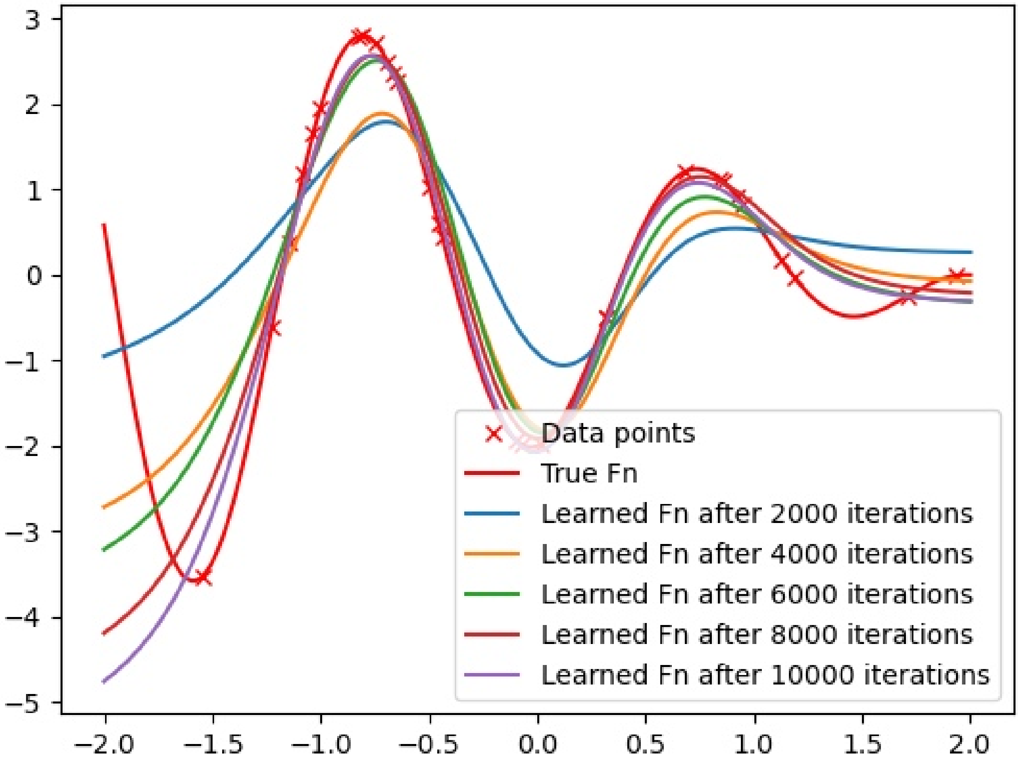
```
import numpy as np
import matplotlib.pyplot as plt

def f_true(x) :
    return (x-2)*np.cos(x*4)

def sigmoid(x) :
    return 1 / (1 + np.exp(-x))

def sigmoid_prime(x) :
    return sigmoid(x) * (1 - sigmoid(x))

K = 10000
alpha = 0.007
N, p = 30, 50
np.random.seed(0)
a0 = np.random.normal(loc = 0.0, scale = 4.0, size = p)
b0 = np.random.normal(loc = 0.0, scale = 4.0, size = p)
u0 = np.random.normal(loc = 0, scale = 0.05, size = p)
theta = np.concatenate((a0,b0,u0))


X = np.random.normal(loc = 0.0, scale = 1.0, size = N)
Y = f_true(X)

def f_th(theta, x) :
    return np.sum(theta[2*p : 3*p] * sigmoid(theta[0 : p] * np.reshape(x,(-1,1)) + theta[p : 2*p]), axis=1)

def diff_f_th(theta, x, index) :
    a = theta[0:p]
    b = theta[p:2*p]
    u = theta[2*p:3*p]

    # (f(X)-Y)
    chain = (f_th(theta,x)-Y[index])

    # Differential respectively
    du = sigmoid(a*x+b) * chain
    da = u*sigmoid_prime(a*x+b)*x * chain
    db = u*sigmoid_prime(a*x+b) * chain

    return np.concatenate((da,db,du))


xx = np.linspace(-2,2,1024)
plt.plot(X,f_true(X),'rx',label='Data points')
plt.plot(xx,f_true(xx),'r',label='True Fn')

for k in range(K) :
    # choose random index
    index = np.random.randint(0,N)

    # SGD
```

```
    theta -= alpha*diff_f_th(theta,X[index],index)

    if (k+1)%2000 == 0 :
        plt.plot(xx,f_th(theta, xx),label=f'Learned Fn after {k+1} iterations')

plt.legend()
# plt.show()
plt.savefig('plot.png')
```



Data points
True Fn
Learned Fn after 2000 iterations
Learned Fn after 4000 iterations
Learned Fn after 6000 iterations
Learned Fn after 8000 iterations
Learned Fn after 10000 iterations
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
3
2
1
0
−1
−2
−3
−4
−5