```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 24 08:13:05 2022

@author: raphaelbailly
"""

import sklearn.datasets
import numpy as np
data = sklearn.datasets.load_digits()

import matplotlib.pyplot as plt

X = data['data']

Y = data['target']

def grad(f, epsilon = 10**(-7)):
    def gradf(x,t):
        n = x.shape[0]
        y = []
        for i in range(n):
            ei = np.zeros(n,float)
            ei[i] = epsilon
            y.append((f(x+ei,t)-f(x,t))/epsilon)
        return(np.array(y))
    return(gradf)




def desc_grad(f, N, param0, data, mu = 0.1):
    param = param0
    gradf = grad(f)
    for i in range(N):
        param = param - mu*gradf(param, data)
        print(i)
    return(param)

def softmax(v):
    w = v-np.max(v)
    w = np.exp(w)
    return(w/w.sum())

def vectorize(Y):
    v = np.zeros((len(Y),10), float)
    for i in range(len(Y)):
        v[i,Y[i]]=1
    return(v)



Param = np.random.randn(650)*0.01

M = Param[:640].reshape(10,64)
b = Param[640:]

Y2 = vectorize(Y)

Data = np.hstack((X,Y2))
Data = Data.reshape(-1)

Input = Data.reshape(-1,74)[:,:64]
```

```
Output = Data.reshape(-1,74)[:,64:]


def likelihood_single(x,y,M,b):
    return(softmax(M@x+b)@y)

def loss(Param, Data):
    M = Param[:640].reshape(10,64)
    b = Param[640:]

    loss = 0

    Input = Data.reshape(-1,74)[:,:64]
    Output = Data.reshape(-1,74)[:,64:]

    for i in range(len(Input)):
        x = Input[i]
        y = Output[i]
        loss = loss - np.log(likelihood_single(x,y,M,b))

    return(loss)

def error(Param, Data):
    M = Param[:640].reshape(10,64)
    b = Param[640:]

    nb_error = 0

    Input = Data.reshape(-1,74)[:,:64]
    Output = Data.reshape(-1,74)[:,64:]

    for i in range(len(Input)):
        x = Input[i]
        y = Output[i]
        z = np.argmax(M@x+b)
        nb_error += (1-y[z])


    return(nb_error)

def divide(Data, p_learn, p_test):
    Data_temp = Data.reshape(-1,74)
    v = np.random.rand(len(Data_temp))
    DataL = Data_temp[v<p_learn]
    DataT = Data_temp[(p_learn<v)*(v<p_learn+p_test)]
    DataV = Data_temp[p_learn+p_test<v]
    return(DataL.reshape(-1), DataT.reshape(-1), DataV.reshape(-1))


def error_vizualisation(Param, Data):
    M = Param[:640].reshape(10,64)
    b = Param[640:]

    nb_error = 0

    Input = Data.reshape(-1,74)[:,:64]
    Output = Data.reshape(-1,74)[:,64:]

    for i in range(len(Input)):
        x = Input[i]
        y = Output[i]
        z = np.argmax(M@x+b)
        if(y[z]<1):
            print(np.argmax(y), z)
```

```
            plt.imshow(x.reshape(8,8), cmap = "Greys")
            plt.show()


    return(nb_error)

DataL, DataT, DataV = divide(Data, 0.1, 0.1)

Param2 = desc_grad(loss, 40, Param, DataL, mu = 0.0001)
```