import numpy as np  
import matplotlib.pyplot as plt  
from scipy.special import expit as f\_act  
  
def init\_net():  
 input\_nodes = 784  
 print('Input the number of hidden neurons:')  
 hidden\_nodes = int(input())  
 out\_nodes = 10  
 print('Input the training speed (0.5):')  
 learn\_speed = float(input())  
 return input\_nodes, hidden\_nodes, out\_nodes, learn\_speed  
  
#3  
def create\_net(input\_nodes, hidden\_nodes, out\_nodes):  
 w\_in2hidden = np.random.uniform(-0.5,0.5,(hidden\_nodes,input\_nodes))  
 w\_hidden2out = np.random.uniform(-0.5,0.5,(out\_nodes,hidden\_nodes))  
 return w\_in2hidden, w\_hidden2out  
  
  
#4  
def net\_output(w\_in2hidden, w\_hidden2out, input\_signal, return\_hidden):  
 input = np.array(input\_signal, ndmin = 2 ).T  
 hidden\_in = np.dot(w\_in2hidden, input)  
 hidden\_out = f\_act(hidden\_in)  
 final\_in = np.dot(w\_hidden2out, hidden\_out)  
 final\_out = f\_act(final\_in)  
 if return\_hidden == 0:  
 return final\_out  
 else:  
 return final\_out, hidden\_out  
  
#5  
def net\_train(target\_list, input\_signal, w\_in2hidden, w\_hidden2out, learn\_speed):  
 targets = np.array(target\_list, ndmin = 2).T;  
 inputs = np.array(input\_signal, ndmin = 2).T;  
 final\_out, hidden\_out = net\_output(w\_in2hidden, w\_hidden2out, input\_signal, 1)  
 out\_errors = targets-final\_out  
 hidden\_errors = np.dot(w\_hidden2out.T, out\_errors)  
  
 w\_hidden2out += learn\_speed\*np.dot((out\_errors\*final\_out\*(1-final\_out)),hidden\_out.T)  
 w\_in2hidden += learn\_speed\*np.dot((hidden\_errors\*hidden\_out\*(1-hidden\_out)),inputs.T)  
  
#6  
def train\_set(w\_in2hidden, w\_hidden2out, learn\_speed):  
 data\_file = open("C:\\mnist\_train.csv", 'r')  
 training\_list = data\_file.readlines()  
 data\_file.close()  
 for record in training\_list:  
 all\_values = record.split(',')  
 # range of input data is scaled from [0.0,255] to [0.001,1.0]  
 inputs = (np.asfarray(all\_values[1:]) / 255.0 \* 0.999) + 0.001  
 targets = np.zeros(10)+0.001  
 # digits 0-9  
 targets[int(all\_values[0])] = 1.0  
 net\_train(targets, inputs, w\_in2hidden, w\_hidden2out, learn\_speed)  
 return w\_in2hidden, w\_hidden2out  
  
#7  
def test\_set(w\_in2hidden,w\_hidden2out):  
 data\_file = open("C:\\mnist\_test.csv", 'r')  
 test\_list = data\_file.readlines()  
 data\_file.close()  
 test = []  
 for record in test\_list:  
 all\_values = record.split(',')  
 # range of input data is scaled from [0.0,255] to [0.001,1.0]  
 inputs = (np.asfarray(all\_values[1:]) / 255.0 \* 0.999) + 0.001  
 out\_session = net\_output(w\_in2hidden,w\_hidden2out, inputs, 0)  
 if int(all\_values[0]) == np.argmax(out\_session):  
 test.append(1)  
 else:  
 test.append(0)  
 test = np.asarray(test)  
 print('Net efficiency % =', (test.sum()/test.size)\*100)  
  
#8  
def plot\_image(pixels:np.array):  
 plt.imshow(pixels.reshape((28,28)),cmap='gray')  
 plt.show()  
  
  
  
input\_nodes, hidden\_nodes, out\_nodes, learn\_speed = init\_net()  
w\_in2hidden, w\_hidden2out = create\_net(input\_nodes, hidden\_nodes, out\_nodes)  
My\_Variant = 5  
for i in range(5):  
 print('Test#',i+1)  
 train\_set(w\_in2hidden, w\_hidden2out, learn\_speed)  
 test\_set(w\_in2hidden, w\_hidden2out)  
  
data\_file = open("C:\\mnist\_test.csv", 'r')  
test\_list = data\_file.readlines()  
data\_file.close()  
all\_values = test\_list[int(My\_Variant-1)].split(',')  
inputs = (np.asfarray(all\_values[1:]) / 255.0 \* 0.999) + 0.001  
out\_session = net\_output(w\_in2hidden, w\_hidden2out, inputs, 0)  
print(np.argmax(out\_session))  
plot\_image(np.asfarray(all\_values[1:]))