LAB 5: HOÀN CHỈNH BÀI TẬP MẠNG NEURON

(SV thực hiện tại lớp)

***

GẮN KẾT DỮ LIỆU TỪ BÀI LAB TỪ 1 ĐẾN 5 ĐỂ HOÀN THÀNH MẠNG NEURON VỜI 02 ĐẦU VÀO VÀ 2 KẾT QUẢ LÀ 0 VÀ 1 (DỰ ĐOÁN MÀU HOA ĐỎ VÀ XANH)

1.1. Thực hiện việc khởi tạo dữ liệu đầu vào và xác định tỷ lệ dịch chuyển của learing_rate =0.1

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

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

# Đạo hàm của hàm sigmoid
def sigmoid_p(x):
    return sigmoid(x) * (1-sigmoid(x))

data = [[3,   1.5, 1],
        [2,   1,   0],
        [4,   1.5, 1],
        [3,   1,   0],
        [3.5, .5,  1],
        [2,   .5,  0],
        [5.5,  1,  1],
        [1,    1,  0]]

#plt.axis([0, 6, 0, 6 ])
#plt.grid()

w1 = np.random.randn()
w2 = np.random.randn()
b = np.random.randn()

learning_rate = 0.1 # tỷ lệ của một bước dịch chuyển trọng số
costs = []
```

1.2. Thiết lập huấn luyện mạng với 5000 bước lập.

```
# chạy vòng lặp với 5000 lần huấn luyện
for i in range(5000):
    ri = np.random.randint(len(data))
    point = data[ri]

    #In dữ liệu cập nhật trong 5000 bước
    #print(point)

    z = point[0] * w1  + point[1] * w2 + b
    pred = sigmoid(z)

    #print("Thông số dự đoán là: {} ".format(pred))
```

1.3. Tính độ chí phí do dự đoán sai, hiệu bình phương của (dự đoán - kết quả)

```
    #tính độ chí phí do dự đoán sai, hiệu bình phương của (dự đoán - kết quả)
    target = point[2]
    cost = np.square(pred - target)
```

1.4.Lan truyền ngược cập nhật thông số khi dự đoán sai.

```
    #Lan truyền ngược cập nhật thông số khi dự đoán sai
    dcost_pred = 2 * (pred - target) # Đạo hàm của cost

    dpred_dz = sigmoid_p(z) # Trả về đạo hàm của hàm Sigmoid(z)

    dz_dw1 = point[0]
    dz_dw2 = point[1]
    dz_db = 1

    dcost_dz = dcost_pred * dpred_dz

    dcost_dw1 = dcost_dz * dz_dw1
    dcost_dw2 = dcost_dz * dz_dw2
    dcost_db = dcost_dz * dz_db
```

1.5. Cập nhật lại thông số w1,w2 và Bias

```
    #Lan truyền ngược cập nhật thông số khi dự đoán sai
    dcost_pred = 2 * (pred - target) # Đạo hàm của cost

    dpred_dz = sigmoid_p(z) # Trả về đạo hàm của hàm Sigmoid(z)

    dz_dw1 = point[0]
    dz_dw2 = point[1]
    dz_db = 1

    dcost_dz = dcost_pred * dpred_dz

    dcost_dw1 = dcost_dz * dz_dw1
    dcost_dw2 = dcost_dz * dz_dw2
    dcost_db = dcost_dz * dz_db

    #cập nhật lại thông số gồm w1,w2,bias

    w1 = w1 - learning_rate * dcost_dw1
    w2 = w2 - learning_rate * dcost_dw2
    b = b - learning_rate * dcost_db
```

1.6. Vẽ sơ đồ chi phí trên tất cả các điểm dữ liệu cứ sau 5000 bước luyện tập của máy.

```
    #cập nhật lại thông số gồm w1,w2,bias

    w1 = w1 - learning_rate * dcost_dw1
    w2 = w2 - learning_rate * dcost_dw2
    b = b - learning_rate * dcost_db

    # in chi phí trên tất cả các điểm dữ liệu cứ sau 5000 bước luyện
    if i % 100 == 0:
        c = 0
        for j in range(len(data)):
            p = data[j]
            p_pred = sigmoid(w1 * p[0] + w2 * p[1] + b)
            c += np.square(p_pred - p[2])
            costs.append(c)

#Hiện biểu đồ tuyến tính có giá trị giảm dần đến và bảo toàn
plt.plot(costs)
```

1.7. Xác định giá trị dự đóa hoa Đỏ và hoa Xanh

```
#Hiện biểu đồ tuyến tính có giá trị giảm dần đến và bảo toàn
plt.plot(costs)

def Du_doan_hoa(length, width):
    z = length * w1 + width * w2 + b
    pred = sigmoid(z)
    if pred < .5:

        print("Hoa có màu Xanh nhé!")
    else:

        print("Màu có màu Đỏ nè")

# test dữ liệu đầu vào và đưa ra thông số kết quả dự đoán

#Du_doan_hoa(1, 1) # ===> Thực tế là màu xanh

#Du_doan_hoa(4.5, 1) # ===> Thực tế là màu đỏ

Du_doan_hoa(4, 1) # ===> Thực nghiệm ko có trong dữ liệu
```

==========HẾT==========