**Bài 20: Neural Networks - Lập trình AI bằng Python**

Mạng nơ-ron là các thiết bị tính toán song song nhằm tạo ra một mô hình máy tính của bộ não. Mục tiêu chính đằng sau là phát triển một hệ thống để thực hiện các tác vụ tính toán khác nhau nhanh hơn các hệ thống truyền thống. Các nhiệm vụ này bao gồm Nhận dạng và Phân loại Mẫu, Ước lượng, Tối ưu hóa và Phân cụm Dữ liệu.

**1. Artificial Neural Networks (ANN) là gì ?**

Mạng nơron nhân tạo (ANN) là một hệ thống tính toán hiệu quả có chủ đề trung tâm được mượn từ sự tương tự của mạng nơron sinh học. ANN cũng được đặt tên là Hệ thống thần kinh nhân tạo, Hệ thống xử lý phân tán song song và Hệ thống kết nối. ANN có được một bộ sưu tập lớn các đơn vị được kết nối với nhau theo một số kiểu để cho phép liên lạc giữa chúng. Những đơn vị này, còn được gọi là nút hoặc nơ-ron, là những bộ xử lý đơn giản hoạt động song song.

Mọi nơron đều được kết nối với nơron khác thông qua một liên kết kết nối. Mỗi liên kết kết nối được liên kết với một trọng số có thông tin về tín hiệu đầu vào. Đây là thông tin hữu ích nhất cho các tế bào thần kinh để giải quyết một vấn đề cụ thể vì trọng lượng thường kích thích hoặc ức chế tín hiệu được truyền đạt. Mỗi nơ-ron đang có trạng thái bên trong được gọi là tín hiệu kích hoạt. Tín hiệu đầu ra, được tạo ra sau khi kết hợp tín hiệu đầu vào và quy tắc kích hoạt, có thể được gửi đến các đơn vị khác.

**2. Cài đặt package:**

Để tạo mạng thần kinh bằng Python, chúng ta có thể sử dụng một gói mạnh mẽ cho mạng thần kinh có tên là NeuroLab. Nó là một thư viện các thuật toán mạng nơ-ron cơ bản với các cấu hình mạng linh hoạt và các thuật toán học cho Python.

**pip** install NeuroLab

Nếu bạn đang sử dụng môi trường Anaconda

conda install -c labfabulous neurolab

**3. Xây dựng Neural Networks**

Trong phần này, chúng ta sẽ xây dựng một số mạng thần kinh trong Python bằng cách sử dụng gói NeuroLab.

**4. Bộ phân loại dựa trên Perceptron**

Perceptron là các khối xây dựng của ANN

**import** matplotlib.pyplot as plt

**import** neurolab as nl

Nhập các giá trị đầu vào. Lưu ý rằng đó là một ví dụ về học có giám sát, do đó bạn cũng sẽ phải cung cấp các giá trị mục tiêu.

input = [[0, 0], [0, 1], [1, 0], [1, 1]]

target = [[0], [0], [0], [1]]

Khởi tạo 2 input và 1 neuron

net = nl.net.newp([[0, 1],[0, 1]], 1)

Bây giờ, training model. Ở đây, mình đang sử dụng quy tắc Delta để đào tạo.

error\_progress = net.train(input, target, epochs=100, show=10, lr=0.1)

visualize kết quả đầu ra và vẽ biểu đồ :

**plt**.figure()

**plt**.plot(error\_progress)

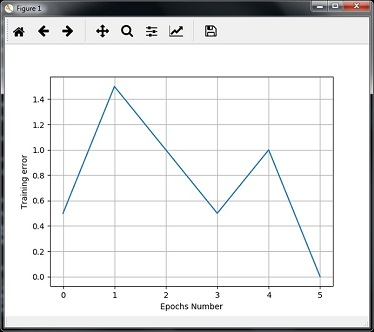
**plt**.xlabel('Number of epochs')

**plt**.ylabel('Training error')

**plt**.grid()

**plt**.show()

Bạn có thể thấy biểu đồ sau cho thấy tiến trình đào tạo bằng cách sử dụng error metric:



**5. Mạng nơ ron một lớp**

Trong ví dụ này, chúng ta đang tạo một mạng nơ-ron một lớp bao gồm các nơ-ron độc lập hoạt động trên dữ liệu đầu vào để tạo ra đầu ra. Lưu ý rằng chúng tôi đang sử dụng tệp văn bản có tên neural\_simple.txt làm đầu vào.

**import** numpy as np

**import** matplotlib.pyplot as plt

**import** neurolab as nl

Load dữ liệu như sau :

input\_data = np.loadtxt(“/Users/admin/neural\_simple.txt')

Sau đây là dữ liệu mà mình sẽ sử dụng. Lưu ý rằng trong dữ liệu này, hai cột đầu tiên là các tính năng và hai cột cuối cùng là nhãn.

array([[2. , 4. , 0. , 0. ],

[1.5, 3.9, 0. , 0. ],

[2.2, 4.1, 0. , 0. ],

[1.9, 4.7, 0. , 0. ],

[5.4, 2.2, 0. , 1. ],

[4.3, 7.1, 0. , 1. ],

[5.8, 4.9, 0. , 1. ],

[6.5, 3.2, 0. , 1. ],

[3. , 2. , 1. , 0. ],

[2.5, 0.5, 1. , 0. ],

[3.5, 2.1, 1. , 0. ],

[2.9, 0.3, 1. , 0. ],

[6.5, 8.3, 1. , 1. ],

[3.2, 6.2, 1. , 1. ],

[4.9, 7.8, 1. , 1. ],

[2.1, 4.8, 1. , 1. ]])

Bây giờ, hãy tách 4 cột này thành 2 cột dữ liệu và 2 nhãn -

data = input\_data[:, 0:2]

labels = input\_data[:, 2:]

Vẽ đồ thị dữ liệu đầu vào bằng các lệnh sau:

**plt**.figure()

**plt**.scatter(data[:,0], data[:,1])

**plt**.xlabel('Dimension 1')

**plt**.ylabel('Dimension 2')

**plt**.title('Input data')

Bây giờ, xác định các giá trị tối thiểu và tối đa cho mỗi thứ nguyên như được hiển thị ở đây -

dim1\_min, dim1\_max = data[:,0].min(), data[:,0].max()

dim2\_min, dim2\_max = data[:,1].min(), data[:,1].max()

Tiếp theo, xác định số lượng tế bào thần kinh trong lớp đầu ra :

nn\_output\_layer = labels.shape[1]

xác định mạng nơ-ron một lớp :

dim1 = [dim1\_min, dim1\_max]

dim2 = [dim2\_min, dim2\_max]

neural\_net = nl.net.newp([dim1, dim2], nn\_output\_layer)

Huấn luyện mạng nơron với số epochs và learning rate như sau:

error = neural\_net.train(data, labels, epochs = 200, show = 20, lr = 0.01)

Tiếp tục visualize và vẽ quá trình huấn luyện :

**plt**.figure()

**plt**.plot(error)

**plt**.xlabel('Number of epochs')

**plt**.ylabel('Training error')

**plt**.title('Training error progress')

**plt**.grid()

**plt**.show()

Sử dụng các điểm dữ liệu thử nghiệm trong bộ phân loại trên

print('\nTest Results:')

data\_test = [[1.5, 3.2], [3.6, 1.7], [3.6, 5.7],[1.6, 3.9]] **for** item **in** data\_test:

print(item, '-->', neural\_net.sim([item])[0])

Bạn có thể tìm thấy kết quả như sau :

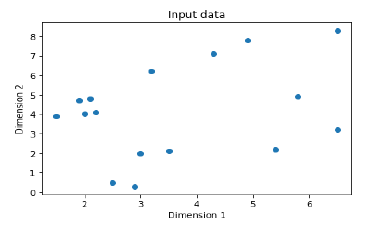
[1.5, 3.2] --> [1. 0.]

[3.6, 1.7] --> [1. 0.]

[3.6, 5.7] --> [1. 1.]

[1.6, 3.9] --> [1. 0.]

Các biểu đồ sau là đầu ra





**6. Mạng nơ ron nhiều lớp**

Trong ví dụ này, mình đang tạo một mạng nơ-ron nhiều lớp bao gồm nhiều hơn một lớp để trích xuất các mẫu cơ bản trong dữ liệu huấn luyện. Mạng nơ-ron nhiều lớp này sẽ hoạt động giống như một bộ hồi quy. Chúng tôi sẽ tạo một số điểm dữ liệu dựa trên phương trình: y = 2x^2+8.

**import** numpy as np

**import** matplotlib.pyplot as plt

**import** neurolab as nl

Tạo một số điểm dữ liệu dựa trên phương trình đã đề cập ở trên

min\_val = -30

max\_val = 30

num\_points = 160

x = np.linspace(min\_val, max\_val, num\_points)

y = 2 \* np.square(x) + 8

y /= np.linalg.norm(y)

Bây giờ, hãy reshape lại tập dữ liệu này như sau:

data = x.reshape(num\_points, 1)

labels = y.reshape(num\_points, 1)

Visualize và vẽ tập dữ liệu đầu vào như sau :

**plt**.figure()

**plt**.scatter(data, labels)

**plt**.xlabel('Dimension 1')

**plt**.ylabel('Dimension 2')

**plt**.title('Data-points')

Bây giờ, hãy xây dựng mạng nơ-ron có hai lớp ẩn với neurolab với mười nơ-ron ở lớp ẩn đầu tiên, sáu ở lớp ẩn thứ hai và một ở lớp đầu ra.

neural\_net = nl.net.newff([[min\_val, max\_val]], [10, 6, 1])

Sau đó ta sẽ sử dụng thuật toán gradient training

neural\_net.trainf = nl.train.train\_gd

Huấn luyện mạng với mục tiêu học tập dữ liệu được tạo như sau :

error = neural\_net.train(data, labels, epochs = 1000, show = 100, goal = 0.01)

Khởi chạy mạng neural train tập dữ liệu training :

output = neural\_net.sim(data)

y\_pred = output.reshape(num\_points)

**plt**.figure()

**plt**.plot(error)

**plt**.xlabel('Number of epochs')

**plt**.ylabel('Error')

**plt**.title('Training error progress')

Mình sẽ vẽ biểu đồ đầu ra thực tế so với dự đoán

**x\_dense** = np.linspace(min\_val, max\_val, num\_points \* 2)

y\_dense\_pred = neural\_net.sim(x\_dense.reshape(x\_dense.size,1)).reshape(x\_dense.size)

plt.figure()

plt.plot(x\_dense, y\_dense\_pred, '-', x, y, '.', x, y\_pred, 'p')

plt.title('Actual vs predicted')

plt.show()

Kết quả :

