**Bài 24: Deep Learning - Lập trình AI bằng Python**

Mạng thần kinh nhân tạo (ANN) nó là một hệ thống tính toán hiệu quả, có chủ đề trung tâm được mượn từ sự tương tự của mạng thần kinh sinh học. Mạng nơ-ron là một loại mô hình cho việc học máy. Vào giữa những năm 1980 và đầu những năm 1990, nhiều tiến bộ quan trọng về kiến trúc đã được thực hiện trong mạng nơ-ron. Trong chương này, bạn sẽ tìm hiểu thêm về Học sâu, một cách tiếp cận của AI.

Học sâu xuất hiện từ sự phát triển bùng nổ về tính toán của một thập kỷ với tư cách là một đối thủ nặng ký trong lĩnh vực này. Do đó, học sâu là một loại học máy cụ thể có các thuật toán được lấy cảm hứng từ cấu trúc và chức năng của não người.

**1. Machine Learning v/s Deep Learning :**

Học sâu là kỹ thuật học máy mạnh mẽ nhất hiện nay. Nó rất mạnh mẽ bởi vì họ học được cách tốt nhất để trình bày vấn đề trong khi học cách giải quyết vấn đề. Dưới đây là bảng so sánh giữa Học sâu và Học máy

**a. Data Dependency**

Điểm khác biệt đầu tiên là dựa trên hiệu suất của DL và ML khi quy mô dữ liệu tăng lên. Khi dữ liệu lớn, các thuật toán học sâu hoạt động rất tốt.

**b. Machine Dependency**

Các thuật toán học sâu cần những cỗ máy cao cấp để hoạt động hoàn hảo. Mặt khác, các thuật toán học máy cũng có thể hoạt động trên các máy cấp thấp.

**c. Feature Extraction**

Các thuật toán học sâu có thể trích xuất các tính năng cấp cao và cố gắng học hỏi từ các tính năng tương tự. Mặt khác, cần có một chuyên gia để xác định hầu hết các tính năng được trích xuất bằng máy học.

**d. Time of Execution**

Thời gian thực thi phụ thuộc vào nhiều tham số được sử dụng trong một thuật toán. Học sâu có nhiều tham số hơn các thuật toán học máy. Do đó, thời gian thực thi của các thuật toán DL, đặc biệt là thời gian đào tạo, nhiều hơn các thuật toán ML. Nhưng thời gian thử nghiệm của thuật toán DL ít hơn thuật toán ML.

**e. Phương pháp tiếp cận để giải quyết vấn đề**

Học sâu giải quyết vấn đề từ đầu đến cuối trong khi học máy sử dụng cách truyền thống để giải quyết vấn đề, tức là chia nhỏ nó thành nhiều phần.

**2. Convolutional Neural Network (CNN)**

Mạng nơ-ron chuyển đổi cũng giống như mạng nơ-ron thông thường vì chúng cũng được tạo thành từ các nơ-ron có trọng số và độ lệch có thể học được. Mạng nơ-ron thông thường bỏ qua cấu trúc của dữ liệu đầu vào và tất cả dữ liệu được chuyển đổi thành mảng 1-D trước khi đưa nó vào mạng. Quá trình này phù hợp với dữ liệu thông thường, tuy nhiên nếu dữ liệu có chứa hình ảnh, quá trình này có thể phức tạp.

CNN giải quyết vấn đề này một cách dễ dàng. Khi xử lý chúng phải tính đến cấu trúc 2D của hình ảnh, điều này cho phép chúng trích xuất các thuộc tính cụ thể cho hình ảnh. Theo cách này, mục tiêu chính của CNN là đi từ dữ liệu hình ảnh thô trong lớp đầu vào đến lớp chính xác trong lớp đầu ra. Sự khác biệt duy nhất giữa NN thông thường và CNN là ở cách xử lý dữ liệu đầu vào và kiểu lớp.

**a. Tổng quan về kiến trúc của CNN :**

Về mặt kiến trúc, các mạng nơ-ron thông thường nhận đầu vào và biến đổi nó thông qua một loạt các lớp ẩn. Mỗi lớp được kết nối với lớp kia với sự trợ giúp của các nơ-ron. Nhược điểm chính của mạng nơ-ron thông thường là chúng không chia tỷ lệ tốt với hình ảnh đầy đủ.

Kiến trúc của CNNs có các nơron được sắp xếp theo 3 chiều gọi là chiều rộng, chiều cao và chiều sâu. Mỗi nơ-ron trong lớp hiện tại được kết nối với một bản vá nhỏ của đầu ra từ lớp trước. Nó tương tự như phủ một bộ lọc 𝑵 × 𝑵 lên hình ảnh đầu vào. Nó sử dụng M bộ lọc để chắc chắn về việc lấy tất cả các chi tiết. Các bộ lọc M này là các công cụ trích xuất tính năng trích xuất các tính năng như cạnh, góc, v.v.

**b. Các lớp được sử dụng để tạo CNN**

Các lớp sau được sử dụng để xây dựng CNN:

* **Input Layer** − Nó lấy dữ liệu hình ảnh thô như nó vốn có.
* **Convolutional Layer** − Lớp này là khối xây dựng cốt lõi của CNN thực hiện hầu hết các tính toán. Lớp này tính toán sự phức tạp giữa các nơ-ron và các bản vá lỗi khác nhau trong đầu vào.
* **Rectified Linear Unit Layer** − Nó áp dụng một chức năng kích hoạt cho đầu ra của lớp trước đó. Nó thêm tính không tuyến tính vào mạng để nó có thể tổng quát hóa tốt cho bất kỳ loại chức năng nào.
* **Pooling Layer** −Việc gộp chung giúp chúng ta chỉ giữ lại những phần quan trọng khi chúng ta tiến triển trong mạng. Lớp gộp hoạt động độc lập trên mọi lát sâu của đầu vào và thay đổi kích thước của nó theo không gian. Nó sử dụng hàm MAX.
* **Fully Connected layer/Output layer**− Lớp này tính toán điểm số đầu ra trong lớp cuối cùng. Kết quả đầu ra có kích thước 𝟏 × 𝟏 × 𝑳, trong đó L là các lớp tập dữ liệu huấn luyện số.

**3. Cài đặt keras:**

Bạn có thể sử dụng Keras, là một API mạng thần kinh cấp cao, được viết bằng Python và có khả năng chạy trên TensorFlow, CNTK hoặc Theno

**pip** install keras

hoặc

conda install –c conda-forge keras

**4. Xây dựng Hồi quy tuyến tính sử dụng ANN**

Trong phần này, bạn sẽ học cách xây dựng một bộ hồi quy tuyến tính bằng cách sử dụng mạng nơron nhân tạo. Bạn có thể sử dụng KerasRegressor để đạt được điều này. Trong ví dụ này, mình đang sử dụng tập dữ liệu giá nhà ở Boston với số 13 cho các bất động sản ở Boston

**import** numpy

**import** pandas

from keras.models **import** Sequential

from keras.layers **import** Dense

from keras.wrappers.scikit\_learn **import** KerasRegressor

from sklearn.model\_selection **import** cross\_val\_score

from sklearn.model\_selection **import** KFold

Load dữ liệu trong đường dẫn local như sau :

dataframe = pandas.read\_csv("/Usrrs/admin/data.csv", delim\_whitespace = True, header = None)

dataset = dataframe.values

chia dữ liệu thành các biến đầu vào và đầu ra, tức là X và Y -

X = dataset[:,0:13]

Y = dataset[:,13]

Định nghĩa model :

**def** **baseline\_model**():

Tạo model như sau :

model\_regressor = Sequential()

model\_regressor.**add**(Dense(13, input\_dim = 13, kernel\_initializer = 'normal',

activation = 'relu'))

model\_regressor.**add**(Dense(1, kernel\_initializer = 'normal'))

Tiếp theo biên dịch model :

model\_regressor.compile(loss='mean\_squared\_error', optimizer='adam')

**return** model\_regressor

chỉnh sửa random seed để có khả năng tái lập

seed = 7

numpy.random.seed(seed)

Đối tượng trình bao bọc Keras để sử dụng trong scikit-learning như một công cụ ước lượng hồi quy được gọi là KerasRegressor. Trong phần này, chúng tôi sẽ đánh giá mô hình này với bộ dữ liệu chuẩn hóa.

estimator = KerasRegressor(build\_fn = baseline\_model, nb\_epoch = 100, batch\_size = 5, verbose = 0)

kfold = KFold(n\_splits = 10, random\_state = seed)

baseline\_result = cross\_val\_score(estimator, X, Y, cv = kfold)

print("Baseline: %.2f (%.2f) MSE" % (Baseline\_result.mean(),Baseline\_result.std()))

Đầu ra của mã hiển thị ở trên sẽ là ước tính hiệu suất của mô hình đối với sự cố đối với dữ liệu không nhìn thấy. Nó sẽ là sai số bình phương trung bình, bao gồm trung bình và độ lệch chuẩn trên tất cả 10 lần đánh giá xác nhận chéo.

**4. Image Classifier: ứng dụng của deep learning**

Convolutional Neural Networks (CNNs) giải quyết một vấn đề phân loại hình ảnh, tức là hình ảnh đầu vào thuộc về lớp nào. Bạn có thể sử dụng thư viện học sâu Keras. Lưu ý rằng chúng tôi đang sử dụng tập dữ liệu đào tạo và thử nghiệm hình ảnh của chó và mèo từ liên kết sau <https://www.kaggle.com/c/dogs-vs-cats/data>.

from keras.models **import** Sequential

Package sau được gọi là Conv2D được sử dụng để thực hiện phép toán tích chập, bước đầu tiên của CNN.

from keras.layers **import** Conv2D

Package sau có tên MaxPoling2D được sử dụng để thực hiện thao tác gộp, bước thứ hai của CNN.

from keras.layers **import** MaxPooling2D

Package sau được gọi là Flatten là quá trình chuyển đổi tất cả các mảng 2D kết quả thành một vectơ tuyến tính dài liên tục.

from keras.layers **import** Flatten

Package sau có tên là Dense được sử dụng để thực hiện kết nối đầy đủ của mạng nơ-ron, bước thứ tư của CNN.

from keras.layers **import** Dense

Bây giờ, hãy tạo một đối tượng của lớp tuần tự.

S\_classifier = Sequential()

bước tiếp theo là codephần tích chập.

S\_classifier.add(Conv2D(32, (3, 3), input\_shape = (64, 64, 3), activation = 'relu'))

Ở đây relu có chức năng chỉnh lưu.

Bây giờ, bước tiếp theo của CNN là thao tác gộp trên các bản đồ đối tượng kết quả sau phần tích chập.

S-classifier.add(MaxPooling2D(pool\_size = (2, 2)))

Chuyển đổi tất cả các hình ảnh gộp thành một vector liên tục bằng cách sử dụng tính năng tâng bốc

**S\_classifier**.add(Flatten())

Tạo một lớp kết nối đầy đủ.

**S\_classifier**.add(Dense(units = 128, activation = 'relu'))

Ở đây, 128 là số đơn vị ẩn. Một thực tế phổ biến là xác định số đơn vị ẩn là lũy thừa của 2.

khởi tạo lớp đầu ra như sau

**S\_classifier**.add(Dense(units = 1, activation = 'sigmoid'))

Biên dịch CNN, ta xây dựng như sau :

**S\_classifier**.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

Ở đây tham số trình tối ưu hóa là chọn thuật toán giảm độ dốc ngẫu nhiên, tham số tổn hao là chọn hàm tổn thất và tham số metrics là chọn chỉ số hiệu suất.

Bây giờ, hãy thực hiện tăng cường hình ảnh và sau đó lắp hình ảnh vào mạng nơ-ron

train\_datagen = ImageDataGenerator(rescale = 1./255,shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255)

training\_set =

train\_datagen.flow\_from\_directory(”/Users/admin/training\_set”,target\_size =

(64, 64),batch\_size = 32,class\_mode = 'binary')

test\_set =

test\_datagen.flow\_from\_directory('test\_set',target\_size =

(64, 64),batch\_size = 32,class\_mode = 'binary')

Điều chỉnh dữ liệu với mô hình mà ta đã tạo

classifier.fit\_generator(training\_set,steps\_per\_epoch = 8000,epochs =

25,validation\_data = test\_set,validation\_steps = 2000)

Ở đây step\_per\_epoch có số lượng hình ảnh đào tạo.

Bây giờ khi mô hình đã được đào tạo, chúng ta có thể sử dụng nó để dự đoán như sau:

from keras.preprocessing import image

test*\_image = image.load\_*img('dataset/single*\_prediction/cat\_*or*\_dog\_*1.jpg',

target\_size = (64, 64))

test*\_image = image.img\_*to*\_array(test\_*image)

test*\_image = np.expand\_*dims(test\_image, axis = 0)

result = classifier.predict(test\_image)

training*\_set.class\_*indices

if result[0][0] == 1:

prediction = 'dog'

else:

prediction = 'cat'