IBM AI Engineering Professional Certificate

# CHƯƠNG 1. Machine Learning with Python

## Giới thiệu về machine learning

Là lĩnh vực con của khoa học máy tính, tập hợp các thuật toán để giải quyết bài toán về dự đoán, phân loại.

Có các phương pháp học tập trong ML là: supervised, unsupervised, semi-supervised, self-supervised, … Trong đó có 2 phương pháp học chính và phổ thông nhất là supervised, unsupervied.

Supervised là phương pháp học có giám sát, nghĩa là khi 1 dữ liệu được đưa vào mô hình thì nó đã được gắn nhãn, hoặc có kết quả rõ ràng, ví dụ: khi đưa dữ liệu vào để máy có thể nhận diện khuôn mặt thì dữ liệu đó đã dược gắn nhãn tương ứng với tên, tuổi, …

Unsupervised là phương pháp học không giám sát, nghĩa là khi 1 dữ liệu được đưa vào mô hình thì mô hình phải cố gắng tìm ra được cấu trúc ẩn bên trong dữ liệu đó, ví dụ: đưa dữ liệu là các loài động vật thì mô hình phải cố gắng tự phân loại các loài vật.

## Bài toán của ML

### Hồi quy

Là dựa vào các thông tin có sẵn để có thể dự đoán một giá trị tiếp theo. Thuật toán phổ biến trong hồi quy là hồi quy tuyến tính.

Hồi quy tuyến tính là tìm ra mối quan hệ giữa các thông tin đầu vào với thông tin đầu ra dựa trên phương trình toán học. Có 2 phương pháp hồi quy tuyến tính là hồi quy tuyến tính đơn giản (chỉ chứa 1 tham số độc lập) và hồi quy tuyến tính bội (chứa nhiều hơn 1 tham số độc lập).

Để đánh giá một mô hình hồi quy thì thường sử dụng các số liệu sau: MAE (Mean absolute error), MSE (Mean Square Error), RSMA (Root Mean Square Error), RAE (Realative Absolute Error), RSA (Relative Square Error).

Tham số đánh giá được sử dụng nhiều nhất là MSE= 1/n \* 2 là khoảng cách bình phương trung bình giữa giá trị thực tế và giá trị mô hình dự đoán.

### Phân loại

Là nhóm các dữ liệu có điểm chung với nhau thành các nhóm. Một số thuật toán phổ biến là hồi quy logistic, k-nearest neighbor, cây quyết định, support vector machine, …

* Hồi quy logistic: là dựa vào các thông tin đã có để đưa ra một dự đoán hữu hạn nhị phân, cụ thể như có hoặc không, đúng hoặc sai, thành công hoặc thất bại, cao hoặc thấp, …

Ví dụ như ta có mối quan hệ như sau: ծ (ƟTX) 🡪 P(y=1|x)

Với mỗi giá trị theta và x, mô hình sẽ cho ra một dự đoán ŷ, và từ đó tính ra xác suất y=1 với mỗi x. Hồi quy logistic sử dụng hàm logistic (sigmoid)

Bước 1: Tạo ngẫu nhiên giá trị theta (Ɵ)

Bước 2: Với mỗi giá trị x từ dữ liệu tính toán ra ŷ (ŷ= ծ (ƟTX))

Bước 3: Tính toán mất mát giữa giá trị được tính bởi mô hình và thực tế (y - ŷ)

Bước 4: Tính toán tổng mất mát giữa mọi giá trị từ dữ liệu cho trước (J())

Bước 5: Thay đổi theta bằng giá trị khác và thực hiện lại Bước 2 đến Bước 4

Thuật toán hồi quy logistic cố gắng tìm ra các giá trị theta sao cho tổng mất mát giữa giá trị mô hình và thực tế thấp nhất (J() min). Khi đạt được đến độ chính xác yêu cầu thì mô hình ngừng lại. Để tối ưu hóa bước 5 thì có thể sử dụng gradient descent để có thể đạt hiệu suất tốt nhất.

Có thể sử dụng các phương pháp thay đổi tham số khác ngoài phương pháp gradient, mục đích của việc thay đổi tham số để tìm ra được theta phù hợp nhất, hàm mất mát giữa giá trị dự đoán và giá trị thực tế là thấp nhất. Đây cũng là bước khó và quan trọng nhất trong hồi quy logistic.

* Support Vector Machine

Đưa dữ liệu đầu vào vào các không gian đa chiều, thuật toán SVM sẽ tạo ra một mặt siêu phẳng bởi phương trình ngăn cách các dữ liệu với nhau. Với không gian 2 chiều gồm x, y thì siêu phẳng là một đường thẳng, với không gian 3 chiều thì siêu phẳng là một mặt phẳng, …

Để đưa dữ liệu vào các chiều không gian cao hơn thì sử dụng các hàm toán học khác nhau như: sigmoid, RBF, đa thức, …

Hàm SVM đạt hiệu quả tốt với các dữ liệu nhiều chiều và tối ưu hóa bộ nhớ, tuy nhiên lại không đạt được kết quả tốt với những dữ liệu lớn từ 1000 dòng và không cung cấp trực tiếp ước tính xác suất. SVM sử dụng tốt cho việc phát hiện chữ viết tay, phát hiện thư rác, phân tích dữ liệu GEN.

* K -Mean Clustering

Thuật toán này nhóm các dữ liệu dựa trên khoảng cách thành các cụm, việc xác định khoảng cách giữa các đối tượng (dữ liệu) bằng các hàm toán học khác nhau, trong đó hàm khoảng cách thông dụng nhất là euclide.

Phương pháp K -Mean sẽ đặt các vị trí k-centroid ngẫu nhiên, từ đó gom các đối tượng vào các k-centroid thành các cụm. Sau đó tính toán lại các vị trí k-centroid bằng cách tính trung bình khoảng cách của các đối tượng trong cụm. Rồi gom các đối tượng lại thành các cụm mới và thực hiện lại các bước tính toán vị trí k -centroid cho đến khi vị trí này không đổi nữa.

Nhược điểm của thuật toán này là phải xác định được trước số cụm của dữ liệu để có thể khởi tạo được bao nhiêu vị trí k -centroid, điều này là không dễ dàng.

## Các thông số để đánh giá mô hình thuật toán phân loại.

Để đánh giá một mô hình ML phân loại cần xác định các thông số sau:

Accuracy, precision, recall, f1 score, ma trận nhầm lẫn, trong đó 2 thông số thường xuyên được sử dụng là accuracy (độ chính xác), precision (độ chính xác khi kết quả dự đoán đúng).

Accuracy = (TP +TN)/ (tổng số số lượng dự đoán/phân loại)

Precision = TP/(TP+FP)

# Chương 2. Introduction to Deep Learning & Neural Networks with Keras

## 2.1 Introduction to Deep Learning

DL là một nhánh nhỏ của ML. Nó bao gồm nhiều lớp (layer), đầu vào là input layer, đầu ra là output layer, giữa 2 lớp đầu vào và đầu ra là các lớp ẩn (hidden layer). Bên trong mỗi lớp sẽ chứa các nút mạng (node). Mỗi node sẽ thực hiện 1 nhiệm vụ, đầu ra của node này sẽ là đầu vào của lớp tiếp theo.

A diagram of a network

Description automatically generated

Hình 0.1

### 2.1.1 Hàm kích hoạt

Trước khi được đưa vào các lớp tiếp theo thì dữ liệu sẽ được đưa qua hàm kích hoạt nhằm giúp mạng có thể học được những biểu diễn phức tạp. Bởi để xử lý các vấn đề phức tạp thì sử dụng mô hình tuyến tính là chưa đủ, và thực tế việc dữ liệu phân bố rất phức tạp, sẽ cần thêm các hàm phi tuyến để có thể biểu diễn tốt hơn.

A graph of a simple and simple model

Description automatically generated

Hình 0.2

Và việc sử dụng hàm kích hoạt sẽ ngăn việc dữ liệu sau mỗi lần qua các node là rất lớn, khi các phương trình có các hàng trăm, hàng triệu tham số thì kết quả của nó có thể là dương vô cùng hoặc âm vô cùng. Vì vậy sử dụng hàm kích hoạt có thể giới hạn giá trị đầu tra ở một khoảng nhất định, như hàm sigmoid là (0;1), ReLU (0,x).

A diagram of a function

Description automatically generated

Hình 0.3

## 2.2 Thuật toán tối ưu

Thuật toán tối ưu là cơ sở để xây dựng mô hình neural network với mục đích "học " được các features (hay pattern) của dữ liệu đầu vào, từ đó có thể tìm 1 cặp weights và bias phù hợp để tối ưu hóa mode (các thông số tối ưu nhất cho phương trình trong node).

Với mỗi giá trị được dự đoán bởi mô hình sẽ có sự khác nhau với giá trị thực tế, khi đó sẽ gọi là mất mát. Hàm mất mát (loss function) được xây dựng để nói lên điều này. Hàm mất mát có công thức tổng quát như sau:

Trong đó y là giá trị thực tế, w\*xi là giá trị mô hình dự đoán, hàm mất mát này là phương trình bậc 2 nên nó sẽ có hình dạng là parabol.

### 2.2.1 Gradient Descent

Tìm giá trị nhỏ nhất của 1 hàm số nào đó bằng cách hàm số đạt giá trị nhỏ nhất khi đạo hàm bằng 0. Đối với các hàm số nhiều biến thì đạo hàm rất phức tạp, thậm chí là bất khả thi. Nên thay vào đó người ta tìm điểm gần với điểm cực tiểu nhất. Gradient Descent có hướng tiếp cận ở đây là chọn 1 nghiệm ngẫu nhiên cứ sau mỗi vòng lặp (hay epoch) thì cho nó tiến dần đến điểm cần tìm. Công thức: wnew = wold – lr \* gradient (w).

Trong đó lr (learning rate) là bước nhảy, gradient là độ dốc của hàm. Với hàm 1 biến thì gradient của hàm là đạo hàm của nó, gradient của hàm nhiều biến là đạo hàm của mỗi biến.

Ví dụ: Xét hàm số f(x)=w2+5sin(w) với đạo hàm f′(w)=2w+5cos(w). Giả sử bắt đầu từ một điểm x= -5, tại vòng lặp thứ t, chúng ta sẽ cập nhật như sau:

wnew = wold – lr \* (2wold+5cos(wold))

chọn w ban đầu = -5 thì gradient của nó là âm, nghĩa là w nằm bên trái của min, lr= 0.1 thì sau 11 bước tính toán gradient của hàm bằng 0, tại đây w không thay đổi và đạt được kết quả tốt nhất.

A graph of a function

Description automatically generated

Hình 0.1

Chọn w ban đầu = 5 thì gradient của hàm là số dương, nên w sẽ nằm bên phải của min, lr= 0.1 thì sau 29 bước gradient của hàm bằng 0, tìm được w tốt nhất

A graph of a function

Description automatically generated

Hình 0.2

Việc lựa chọn learning rate và giá trị ban đầu rất quan trọng trong các bài toán thực tế. Việc lựa chọn giá trị phụ thuộc nhiều vào từng bài toán và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất.

Quy tắc cần nhớ: luôn luôn đi ngược hướng với đạo hàm.

Với các hàm nhiều biến thì phương pháp gradient có dạng như sau:

A graph of a graph with a rainbow colored curve

Description automatically generated with medium confidence

Hình 0.3

### 2.2.2 Lan truyền ngược

Là phương pháp sử dụng tính toán mất mát giữa giá trị dự đoán và giá trị thực tế để cập nhật lại các giá trị w1, w2, b1, b2 cho đến khi mất mát là bé nhất.

A diagram of a diagram

Description automatically generated

Hình 0.4

Để cập nhật các giá trị thì ta sử dụng phương pháp gradient descent cho từng giá trị (w1, w2, b1, b2). Ví dụ w2:

W2new=w2old – lr \* gradient (E(w))

A white board with blue text

Description automatically generated

Hình 0.5

Để tìm ra gradient giữa E (hàm mất mát) với w2 thì ta sử dụng đạo hàm chuỗi. w2 nằm trong z2, z2 nằm trong a2, a2 nằm trong E nên suy ra gradient (E(w2)) là:

A math equations on a white background

Description automatically generated

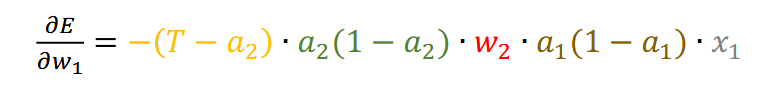
Hình 0.6

Tương tự với gradient của các trọng số b2, w1, b1:

A number of letters and numbers

Description automatically generated with medium confidence

Hình 0.7



Hình 0.8

A number of mathematical equations

Description automatically generated with medium confidence

Hình 0.9

Với các giá trị w1, b1, w2, b2 chọn ngẫu nhiên thì sau số lần lặp chúng ta yêu cầu (vd: epoch= 100) hoặc giá trị mất mát đạt đến ngưỡng (E<0,001) thì dừng lại.

### 2.2.3 Hàm tối ưu Adam

Hàm tối ưu đang được sử dụng nhiều và phổ biến nhất hiện nay.

…….

## 2.3 Thư viện học sâu Keras

Các thư viện phổ biến trong học sâu là Tensorflow, PyTorch, Keras. Tensorflow và Keras được cung cấp bởi Google, Keras là phiên bản thu gọn nhưng vẫn đầy đủ cho những người bắt đầu làm quen để có thể tiếp đến các công cụ cao cấp hơn.

Để sử dụng Keras thì cũng cần cài đặt môi trường Tensorflow và import các thư viện của Keras.

|  |
| --- |
| import keras  from keras.models import Sequential  from keras.layers import Dense |

Thư viện Keras hỗ trợ tạo cấu trúc DL bằng câu lệnh add(Dense()):

|  |
| --- |
| def regression\_model():  # create model  model = Sequential()  model.add(Dense(50, activation='relu', input\_shape=(n\_cols,)))  model.add(Dense(50, activation='relu'))  model.add(Dense(1))    # compile model  model.compile(optimizer='adam', loss='mean\_squared\_error')  return model |

Mô hình học sâu này có 2 lớp ẩn, mỗi lớp ẩn có 50 node, sau mỗi lớp sẽ sử dụng hàm kích hoạt ReLU, đầu vào gồm n\_cols (n dữ liệu đầu vào), đầu ra là 1 dữ liệu ra, sử dụng hàm tối ưu Adam, và hàm mất mát là MSE.

Để tránh việc overfitting (là mô hình có thể nhận biết, tính toán chính xác với những dữ liệu trong tập dữ liệu, nhưng đối với các dữ liệu mới thì mô hình không đạt được kết quả tốt) thì thư viện Keras hỗ trợ phân chia tập dữ liệu thành bộ huấn luyện và bộ kiểm tra theo tỉ lệ nhất định:

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(predictors, target, test\_size=0.3, random\_state=42) |

Chia tập dữ liệu để đự đoán và kết quả theo tỉ lệ 7/3 (bộ dữ liệu test có kích thước 3/10 của tập dữ liệu)

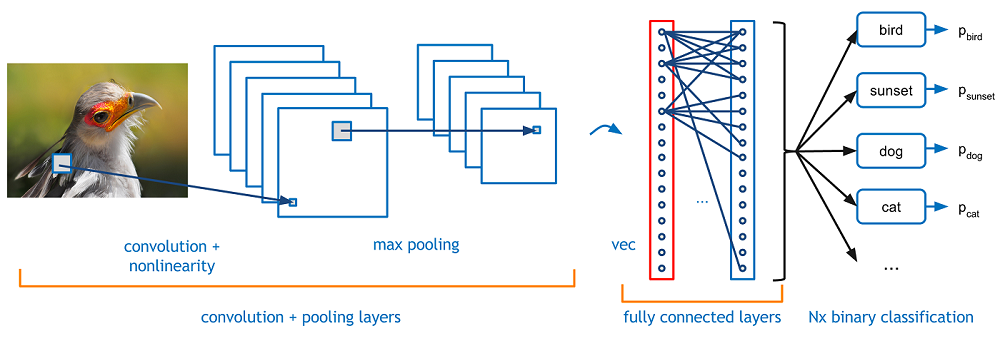
Để chạy thử nô hình phân loại với thư viện Keras như sau:

|  |
| --- |
| # build the model  model = classification\_model()  # fit the model  model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=10, verbose=2)  # evaluate the model  scores = model.evaluate(X\_test, y\_test, verbose=0) |

Tham khảo chi tiết thêm tại trang chủ Keras: <https://keras.io/api/>

## 2.4 Mạng lưới thần kinh tích chập (Convolutional Neural Network -CNN)

CNN được sử dụng chủ yếu cho bộ dữ liệu đầu vào là hình ảnh, CNN là giải pháp tốt nhất để giải quyết các vấn đề liên quan đến hình ảnh như nhận dạng, phát hiện đối tượng.



Hình 0.1

CNN bao gồm một loạt các lớp: lớp tích chập, hàm kích hoạt, hàm pooling cũng như lớp kết nối đầy đủ trước khi kết quả được đưa ra.

Đầu vào của CNN được gọi là Tensor (m\*n\*c): với các hình ảnh chỉ có 1 thang màu là xám thì sẽ là m\*n, các hình ảnh màu sẽ là m\*n\*3 (rbg). Khi một tensor được đưa vào mô hình, mô hình sẽ thực hiện tích chập:

A diagram of a program

Description automatically generated with medium confidence

Hình 0.2

Kết quả sau mỗi đầu ra là một hàm kích hoạt, hàm kích hoạt phổ biến hiện nay là hàm ReLU.

A graph with a blue line

Description automatically generated

Hình 0.3

Hàm này trả về giá trị 0 nếu đầu vào là giá trị âm, và trả về chính nó nếu là giá trị dương. Hàm kích hoạt này không làm mất độ dốc như hàm sigmoid (gradient = 0).

Tiếp theo là hàm tổng hợp. Hàm này giảm bớt kích thước của tensor khi đã đi qua hàm kích hoạt nhưng vẫn giữ được các đặc điểm của tensor, chỉ giữ lại những đặc trưng nổi bật, có 2 phương pháp đó là hàm tổng hợp trung bình hoặc hàm tổng hợp lớn nhất.

A diagram of a number

Description automatically generated

Hình 0.4

Cuối cùng sẽ chuyển đổi các tensor thành các node để sẵn sàng cho việc phân loại, tính toán, …

Để sử dụng mạng tích chập với thư viện Keras ta import các thư viện cần thiết:

|  |
| --- |
| import keras  from keras.models import Sequential  from keras.layers import Dense  from keras.utils import to\_categorical  from keras.layers.convolutional import Conv2D # to add convolutional layers  from keras.layers.convolutional import MaxPooling2D # to add pooling layers  from keras.layers import Flatten # to flatten data for fully connected layers |

Để tạo cấu trúc mạng nhân tạo sử dụng câu lệnh sau:

|  |
| --- |
| def convolutional\_model():    # create model  model = Sequential()  model.add(Conv2D(16, (5, 5), strides=(1, 1), activation='relu', input\_shape=(28, 28, 1)))  model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2, 2)))    model.add(Flatten())  model.add(Dense(100, activation='relu'))  model.add(Dense(num\_classes, activation='softmax'))    # compile model  model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  return model |

Trong đó mạng tích chập có 16 bộ lọc, mỗi bộ lọc có kích thước 5x5, trượt theo các hướng ngang và dọc là 1 đơn vị, hàm kích hoạt ReLU, đầu vào có kích thước 28x28x1 (là hình ảnh màu xám có chiều dài chiều rộng là 28x28).

Hàm tổng hợp sử dụng là maxpooling, kich thước 2x2, trượt 2 đơn vị. Đầu ra của lớp tích chập sau cùng là 1 lớp kết nối đầy đủ có 100 node. Cuối cùng kết qảu đầu là là num\_class (số lượng đầu ra tùy thuộc bài toán, ví dụ xác định chữ số thì num\_class = 10 vì có 10 chữ số từ 0-9), sử dụng hàm kích hoạt softmax, Hàm tối ưu hóa là hàm Adam, hàm mất mát là categorical\_crossentropy để tính độ chính xác của mô hình.

Để chạy thử mô hình này ta sử dụng câu lệnh:

|  |
| --- |
| model = convolutional\_model()  # fit the model  model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=10, batch\_size=200, verbose=2)  # evaluate the model  scores = model.evaluate(X\_test, y\_test, verbose=0)  print("Accuracy: {} \n Error: {}".format(scores[1], 100-scores[1]\*100)) |

Ngoài ra Recurrent Neural Network (RNN) có thể vừa nhận đầu vào mới và nhận thêm đầu vào từ việc lấy đầu ra của node trước. Mô hình này xử lý tốt trong những trường hợp như xử lý cảnh trong video, bởi vì các cảnh quay không phải độc lập, nó có mối liên hệ với nhau, các cảnh quay mới có liên hệ với cảnh quay đã diễn ra, xử lý ảnh để tạo ra một bức ảnh mới, …

### 2.4.1 Autoencoder

A diagram of a machine learning

Description automatically generated

Hình 0.5

Thuật toán này sử dụng mạng lưới thần kinh nhân tạo để xây dựng. Nó tự động mã hóa và giải nén dữ liệu. Ứng dụng của Autoencoder là khử nhiễu dữ liệu, giảm kích thước dữ liệu bằng cách nén nó lại, tái tạo dữu liệu. Đây là mô hình mạng thần kinh không giám sát. Nó sử dụng lan truyền ngược (sử dụng đầu ra để làm dữ liệu cho đầu vào mới). Một mô hình khá phổ biển nữa là Restricted Boltzmann Machines (RBMs).

# Chương 3. Introduction to Computer Vision and Image Processing

## 3.1