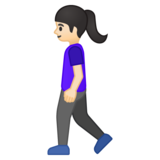
# Hello\_World\_Machine\_Learning

# Bài Tensorflow 1 : Chương trình Hello World trong Machine learning(ML).

#### 1. Giới thiệu.

Vi du : Một chương trình bằng ngôn ngữ lập trình truyền thống nhận diện trạng thái hoạt động trong quá trình giám sát tập luyện thể thao. Bạn cho biết tốc độ của người luyện tập và chương trình sẽ tính trạng thái hoạt động tương ứng thông qua câu lệnh điều :




|  |  |
| --- | --- |
|  | Code : |
|  |  |
|  | **if**(speed &**lt**; 4){ |
|  | status=WALKING; |
|  | } **else** **if**(speed &**lt**; 12){ |
|  | status=RUNNING; |
|  | } **else** { |
|  | status=BIKING; |
|  | } |

#### 2. Mô hình xây dựng ứng dụng truyền thống:



|  |  |
| --- | --- |
|  | Bạn **s**ẽ định nghĩa rules và đưa data vào , chương **tr**ình **s**ẽ tính kết quả tương ứng |

#### 3. Mô hình machine learning:



Thay vì định nghĩa rules , bạn cung cấp tập kết quả (hay còn gọi là labels) tương ứng với data đầu vào . Máy sẽ suy luận ra rules thể hiện mối quan hê giữa data input va label.



Chúng ta thu thập nhiều data/label đủ để máy có thể du đoán chính xác từng trạng thái tương ứng như : đi bộ, chạy, đạp xe …

Trong lập trình truyền thống, code sẽ được dịch thành bin (mã máy) còn được gọi là chương trình ( program). Trong machine learning, kết quả có được từ data input (features) và lables được gọi là model



Bạn sẽ cung cấp dữ liệu và model sẽ dùng những rules từ quá trình training để cho ra kết quả dự đoán đầu ra, vi dụ như : dữ liệu cung cấp trông có vẻ như walking , biking…..

Phần kế tiếp, bạn sẽ coding và xây dựng một model đơn giản “Hello World” mà hầu hết ngữ cảnh machine learning đều dùng.

#### 4. Tạo Machine Learned Model :



Nhận xét :  
X tăng 1 thì Y tăng 3  
khi X = 0 , Y = 1

Suy luận ra : Y = 3\*X +1 : đó chính xác là model mà bạn cần code để đạt được mối tương quan này.

Ta sẽ dùng ngôn ngữ Python va thư viện Tensorflow Keras để xây dưng model:

**a.** Phần import để nạp thư viện Tensorflow cần dùng , thư viện numpy giúp biến input data thành một list một cách dễ dàng và nhanh chóng.

Framework dùng cho định nghĩa 1 mạng huấn luyện (neural network – NN) là 1 tập sequential layers được gọi là Keras.

|  |  |
| --- | --- |
|  | **import** tensorflow **as** tf |
|  | **import** numpy **as** np |
|  | **from** tensorflow **import** keras |

**b.** Định nghĩa một mang NN: bước này ta sẽ tạo ra một mạng đơn giản gồm 1 layer có 1 neuron , input có 1 gía trị :

|  |  |
| --- | --- |
|  | model = tf.keras.Sequential([keras.layers.Dense(units=1, input\_shape=[1])]) |

Sequantial() là thể hiện việc các layer được xây dựng theo đúng thứ tự trong. Phần tử đầu tiên của list thể hiện kết nối giưa input layer và layer tiếp theo, các phần tử tiếp theo của list thể hiện kết nối của các layer tiếp theo.

Để mạng hoạt động cần phải chỉ ra 2 hàm : loss và optimizer

Như đã biết hàm ( function ) là mối quan hệ giữa X, Y , vidu : y = 2x – 1

Khi máy tính học và đoán … y = 10x + 9

Hàm loss có nhiệm vụ đo đạc sự sai khác giữa giá trị đoán được với giá trị đã biết trước để đánh giá việc tính toán (đoán) là tốt hay xấu

Kế tiếp , hàm optimizer sẽ đoán 1 kết quả khác dựa trên kết quả phân tích của hàm loss bằng cách làm giảm sự sai khác nhỏ xuống, vi du : y = 5x + 4 , giá tri này vẫ chưa thực sự tốt.

Model hình sẽ lặp tiến trình này sau một số lần epochs ( 1 epoch là một lần “duyệt” qua hết số lượng mẫu trong tập huấn luyện “mean squared error” ( mse ) (sai số toàn phương trung bình của một phép ước lượng là trung bình của bình phương các sai số) được dùng để tính hàm loss

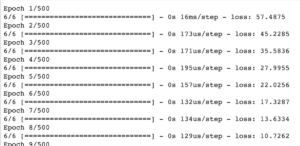
Giải thuật stochastic gradient descent (sgd) được dùng để tính hàm optimizer

|  |  |
| --- | --- |
|  | model.compile(optimizer='sgd', loss='mean\_squared\_error') |

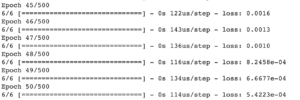
**c.**Phần cung cấp data:  
Python dùng numpy cung cấp dữ liệu dạng mảng ( array) thong qua hàm np.array[]  
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)  
ys = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)Suy luận

#### 5. Huấn luyện mạng :

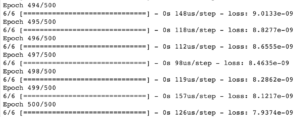
Quá trình huấn luyện mạng để học mối quan hệ giữa X và Y thông qua gọi hàm model.fit như một vòng lặp : đoán kết quả , đo lường loss ( tốt/xấu), optimizer đoán 1 kết quả khác , lặp lại 1 số epoch lần như vi dụ chạy demo bên dưới :  
model.fit(xs, ys, epochs=500)

Trong vài epochs đầu , giá trị loss rất cao :  


Trong quá trình training , giá tri loss giảm dần :



Tại thời điểm quá trình huấn luyện kết thúc , giá trị loss là rất nhỏ :



Không nhất thiết phải là 500 epochs, bạn có thể thay đổi tuỳ theo kinh nghiệm mà bạn có được , như vi dụ trên , epochs = 50 là đủ để có giá trị loss đủ nhỏ.

#### 6. Dùng model :

Bay giờ , bạn đã có model được huấn luyện để học mối quan hệ giữa X và Y , bạn dùng hàm model.predict , vi dụ : X= 10 thì Y = ?

|  |  |
| --- | --- |
|  | **print**(**model**.predict([10.0])) |

Kết là 30.99733 , gần giống với công thức Y = 3X + 1 với X = 10.

Kết luận : Dù tin hay không bạn đã đi qua hầu hết các concepts về machine learning mà bạn có thể áp dụng trong các ngữ cảng phức tạp hơn. Trong ví dụ trên , bạn đã hiểu được bằng cách nào đển huấn luyện 1 mạng để tìm mối quan hệ giữa 2 tập số. Bạn định nghĩa mạng gồm 1 tập các lớp layers ( vi dụ này là 1 ) chứa nhiều neurons ( vi dụ này là 1 ) , sau dó bạn huấn luyện thông qua hàm loss và optimizer.

Tập hợp gồm 1 mạng , hàm loss và hàm optimizer điều khiển quá trình đoán mối quan hệ giữa những tập số , đo lường hiệu chỉnh sai số , tạo ra thông số mới cho giá trị mới , sao cho kết quả là gần đúng nhất.

#### 7. Minh hoạ hình ảnh 1 vòng quá trình forward/backward huấn luyện :

