**Linear Regression/Ordinary least squares (Hồi Quy Tuyến Tính)**

Giải thích: Linear hay tuyến tính hiểu một cách đơn giản là thẳng, phẳng. Trong không gian hai chiều, một hàm số được gọi là tuyến tính nếu đồ thị của nó có dạng một đường thẳng. Trong không gian ba chiều, một hàm số được goi là tuyến tính nếu đồ thị của nó có dạng một mặt phẳng. Trong không gian nhiều hơn 3 chiều, khái niệm mặt phẳng không còn phù hợp nữa, thay vào đó, một khái niệm khác ra đời được gọi là siêu mặt phẳng (*hyperplane*). Các hàm số tuyến tính là các hàm đơn giản nhất, vì chúng thuận tiện trong việc hình dung và tính toán.

**Giới thiệu:**

Một căn nhà rộng  , có  phòng ngủ và cách trung tâm thành phố  **km** có giá là bao nhiêu. Giả sử chúng ta đã có số liệu thống kê từ 1000 căn nhà trong thành phố đó, liệu rằng khi có một căn nhà mới với các thông số về diện tích, số phòng ngủ và khoảng cách tới trung tâm, chúng ta có thể dự đoán được giá của căn nhà đó không? Nếu có thì hàm dự đoán  sẽ có dạng như thế nào. Ở đây  là một vector hàng chứa thông tin input,  là một số vô hướng (scalar) biểu diễn output (tức giá của căn nhà trong ví dụ này).

Một cách đơn giản nhất, chúng ta có thể thấy rằng:

* Dện tích nhà càng lớn thì giá nhà càng cao
* Số lượng phòng ngủ càng lớn thì giá nhà càng cao
* Càng xa trung tâm thì giá nhà càng giảm

Một hàm số đơn giản nhất có thể mô tả mối quan hệ giữa giá nhà và 3 đại lượng đầu vào là:

Trong đó,  là các hằng số, còn được gọi là **bias tức hệ số tự do**. Mối quan hệ  bên trên là một mối quan hệ tuyến tính (*linear*). Bài toán chúng ta đang làm là một bài toán thuộc loại regression. Bài toán đi tìm các hệ số tối ưu chính vì vậy được gọi là bài toán Linear Regression.

**Dạng của Linear Regression**

Trong đó:

* **w = {}** là vector (cột) cần phải tối ưu.
* là vector (hàng) dữ liệu đầu vào, số **1** được đặt ở đầu để phép tính đơn giản và thuận tiện hơn cho việc tính toán.

**Sai số dự đoán:**

Ta mong muốn rằng sự sai khác  giữa giá trị thực  và giá trị dự đoán là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

**=**

Trong đó hệ số để thuận tiện cho việc tính toán (khi tính đạo hàm thì số sẽ bị triệt tiêu). Còn vì , việc ta lấy bình phương cũng là để thuận tiện trong việc đạo hàm.

**Hàm mất mát (cost function/loss funtion)**

Điều tương tự xảy ra với tất cả các cặp (input, outcome) với **N** là số dữ liệu quan sát được. Điều chúng ta muốn, tổng sai số là nhỏ nhất, tương đương với việc tìm **w** để hàm số sau đạt giá trị nhỏ nhất:

Hàm số  được gọi là **hàm mất mát** (cost function/loss function) của bài toán Linear Regression. Chúng ta luôn mong muốn rằng sự mất mát (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số  sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của  làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là điểm tối ưu (optimal point), ký hiệu:

Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát . Đặt là một vector cột chứa tất cả các output của training data. là ma trận dữ liệu đầu vào (mở rộng) mà mỗi hàng của nó là một điểm dữ liệu. Khi đó hàm số mất mát  được viết dưới dạng ma trận đơn giản hơn:

Với là Euclidean norm (chuẩn Euclid, hay khoảng cách Euclid), nói cách khác là tổng bình phương mỗi phần tử của vector **z**. Tới đây, ta đã có một mạng đơn giản của hàm mất mát được viết như phương trình **.**

**Nghiệm cho bài toán Linear Regression**

Để tìm nghiệm cho bài toán tối ưu là giải phương trình đạo hàm (gradient) bằng **0**.

Đạo hàm theo của hàm mất mát là:

Phương trình đạo hàm bằng tương đương với:

Ký hiệu tức là đặt bằng   
Nếu ma trận vuông khả nghịch (non-singular hay invertible) thì phương trình có nghiệm duy nhất: .

Nếu ma trận vuông không khả nghịch (có định thức bằng 0), thì phương trình vô nghiệm hoặc vô số nghiệm. Khi đó ta sẽ xử dụng khái niệm **pseudo inverse** (giả nghịch đảo)  **(A dagger)**. Là trường hợp tổng quát của nghịch đảo khi ma trận không khả nghịch, hoặc không vuông.

Với khái niệm giả nghịch đảo, điểm tối ưu bài toán Linear Regression có dạng:

**Ví dụ trên Python:**

Ta sẽ lấy ví dụ đơn giản về giải về việc giải bài toán Linear Regression trong Python. Ta cũng sẽ so sánh nghiệm của bài toán khi giải theo phương trình và nghiệm tìm được khi dùng thư viện ***scikit-learn*** của Python. Để đơn giản ta sẽ dữ liệu với đầu vào chỉ có 1 chiều (1 giá trị)

Bài toán dự đoán cân nặng của một bạn trong lớp ĐHCN1A dựa vào chiều cao của bạn ấy (Trên thực tế, tất nhiên là không, vì cân nặng còn phụ thuộc vào nhiều yếu tố khác nữa, thể tích chẳng hạn).

|  |  |  |
| --- | --- | --- |
| Họ và Tên | Chiều Cao | Cân Nặng |
| Trịnh Đình Phúc | 168 | 50 |
| Nguyễn Đình Thái | 169 | 55 |
| Nguyễn Đức Tiến | 180 | 62 |
| Trần Trung Hiếu | 176 | 64 |
| Trịnh Văn Bình | 161 | 53 |
| Lê Thị Hoài Thuận | 163 | 46 |
| Trần Thanh Phương | 172 | 65 |
| Lê Huy Hoàng | 170 | 58 |
| Bùi Lý Hải Đăng | 173 | 73 |
| Huỳnh Diệp Phụng | 163 | 55 |
| Lê Hùng Phú | 173 | 74 |
| Huỳnh Quốc Đạt | 169 | 52 |
| Phạm Hữu Danh | 172 | 71 |
| Nguyễn Duy Thanh Tùng | 172 | ? |

Để kiểm tra độ chính xác của model tìm được, chúng ta sẽ giữ lại hàng tên “Nguyễn Duy Thanh Tùng” để kiểm thử, các hàng còn lại được sử dụng để huấn luyện (train) model.

**Hiển thị dữ liệu trên đồ thị:**

*#Trước tiên, chúng ta cần có hai thư viện numpy cho đại số tuyến tính và matplotlib cho việc vẽ hình.***import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
*# chiều cao (cm)*X = np.array([[168,169,180,176,161,163,172,170,173,163,173,169,172]]).T  
*# cân nặng (kg)*y = np.array([[ 50 ,55 ,62, 64, 53 , 46, 65, 58, 73, 55, 74, 52, 71]]).T  
*# Visualize data*plt.plot(X, y, **'ro'**)  
plt.axis([140, 190, 45, 75])  
plt.xlabel(**'Chiều cao (cm)'**)  
plt.ylabel(**'Cân nặng (kg)'**)  
plt.show()

***Kết quả:***



*Hình 1: hiển thị dữ liệu về chiều cao & cân nặng trên đồ thị*

**Nghiệm theo công thức:**

Công thức: **Can\_nang =** **w\_1\*(chieu\_cao) + w\_0**

Tiếp theo, ta sẽ tính toán các hệ số **w\_1** và **w\_0** dựa vào công thức

*# xây dựng X feature & reshape*one = np.ones((X.shape[0], 1))  
Xbar = np.concatenate((one, X), axis = 1)  
*# tính các trọng số A,b,w*A = np.dot(Xbar.T, Xbar) *#np.dot tích trong 2 ma trận*b = np.dot(Xbar.T, y)  
w = np.dot(np.linalg.pinv(A), b) *#pseudo-inverse  
# chuẩn bị cho fitting line*w\_0 = w[0][0]  
w\_1 = w[1][0]  
x0 = np.linspace(145,190,2) *#chia ra 2 khoảng từ 145 đến 190 (vì chiều cao nằm trong khoảng đó)*y0 = w\_0 + w\_1\*x0  
print(**"Phương trình sau khi học được là: y ={}x {}"**.format(w\_1,w\_0))



*Hình 1: mô hình linear regression sau khi học xong*

Nhận xét: từ đồ thị trên ta thấy đường màu xanh nằm khá gần với các đường màu đỏ, tức mô hình Linear Regression hoạt động tốt với tập dữ liệu training. Tiếp theo, ta sẽ dự đoán cân nặng của bạn

y1 = w\_1\*172 + w\_0  
print(**"Phương trình sau khi học được là: y ={}x {}"**.format(w\_1,w\_0))  
print( **u'Chiều cao của Thanh Tùng là 172, cân nặng dự đoán: %.2f (kg), cân nặng thực tế là: 60 (kg)'** %(y1) )

*Kết quả:*

Chiều cao của Thanh Tùng là 172, cân nặng dự đoán: 62.16 (kg), cân nặng thực tế là: 60 (kg)

Nhận xét: kết quả dự đoán tương đối chính xác.

**Hạn chế của Linear Regression**

Hạn chế đầu tiên của Linear Regression là nó rất nhạy cảm với nhiễu (sensitive to noise). Trong ví dụ về mối quan hệ giữa chiều cao và cân nặng bên trên, nếu có chỉ một cặp dữ liệu nhiễu (155 cm, 80kg) thì kết quả sẽ sai khác đi rất nhiều. Xem hình dưới đây:



*Hình 1: mô hình linear regression khi gặp nhiễu*

Vì vậy, trước khi thực hiện Linear Regression, các nhiễu (outlier) cần phải được loại bỏ. Bước này được gọi là tiền xử lý (pre-processing).

Hạn chế thứ hai của Linear Regression là nó không biễu diễn được các mô hình phức tạp. Mặc dù trong phần trên, chúng ta thấy rằng phương pháp này có thể được áp dụng nếu quan hệ giữa outcome và input không nhất thiết phải là tuyến tính, nhưng mối quan hệ này vẫn đơn giản nhiều so với các mô hình thực tế, những mô hình có các hàm như **.**

TÓM LẠI LINEAR REGRESSION LÀ MÌNH ĐI TÌM MỘT CÁI ĐƯỜNG NÀO ĐÓ SAO CHO TỔNG KHOẢNG CÁCH CỦA CÁC ĐIỂM TỚI CÁI ĐƯỜNG NÀY LÀ NHỎ NHẤT, TỨC LÀ CÁI ĐƯỜNG NÓ FIT CÁC ĐIỂM NÀY