Linear Regression

Trong bài viết, mình sẽ giới thiệu một trong những thuật toán cơ bản nhất của Machine Learning. Đây là thuật toán Linear Regression (Hồi Quy Tuyến Tính) thuộc nhóm Supervised learning (Học có giám sát). Hồi quy tuyến tính là một phương pháp rất đơn giản nhưng đã được chứng minh được tính hữu ích cho một số lượng lớn các tình huống. Trong bài viết này, bạn sẽ khám phá ra chính xác cách thức tuyến tính làm việc như thế nào. Trong việc phân tích dữ liệu, bạn sẽ tiếp xúc với thuật ngữ "Regression" (Hồi quy) rất thường xuyên. Trước khi đi sâu vào Hồi quy tuyến tính, hãy tìm hiểu khái niệm Hồi quy trước đã. Hồi quy chính là một phương pháp thống kê để thiết lập mối quan hệ giữa một biến phụ thuộc và một nhóm tập hợp các biến độc lập. Ví dụ :

```
Tuổi = 5 + Chiều cao * 10 + Trọng lượng * 13
```

Ở đây chính ta đang thiết lập mối quan hệ giữa Chiều cao & Trọng lượng của một người với Tuổi của anh/cô ta. Đây là một ví dụ rất cơ bản của Hồi quy.

Table of Contents

1 Giới Thiệu

Quay lại ví dụ được nêu ở đầu bài: một người cao $x_1 cm,$ có $x_2 kg$

trọng lượng thì bao nhiêu tuổi. Giả sử chúng ta đã có số liệu thống kê từ 1000 người trong thành phố, liệu rằng khi có một người mới với các thông số về chiều cao, cân nặng, chúng ta có thể dự đoán được giá của căn nhà đó không? Nếu có hàm dự đoán y=f(x) sẽ có dạng như thế nào. Ở đây $\mathbf{x} = [x_1, x_2]$ là vector hàng chưa thông tin input, y là một số vô hướng biểu diễn output(tức là tuổi của người đó).

$$y \approx f(\mathbf{x}) = \hat{y}$$

Trong đó, w_0, w_1, w_2 là các hằng số, $w0$ còn được gọi là bias. Mối quan hệ $y \approx f(\mathbf{x})$ bên trên là môi quan hệ tuyến tính. Bài toán chúng ta đang làm thuộc loại regression. Cuối cùng, bài toán đi tìm hệ số tối ưu w_1, w_2, w_0 chính vì vậy được gọi là bài toán Linear Regression.

2 Phân tích toán học

2.1 Dạng của Linear Regression

$$\mathrm{f(x)} = \mathrm{w_0} + \mathrm{w_1 x_1} + \mathrm{w_2 x_2} + \cdots + \mathrm{w_n x_n} \qquad (1)$$

Trong phương trình (1) phía trên, nếu chúng ta đặt $\mathrm{w} = [\mathrm{w_0}, \mathrm{w_1}, \mathrm{w_2}, \mathrm{w_3}]^{\mathrm{T}} =$ là vector (cột) hệ số cần phải tối ưu và $\bar{\mathrm{x}} = [1, \mathrm{x_1}, \mathrm{x_2}, \mathrm{x_3}]$ (đọc là x bar trong tiếng Anh) là vector (hàng) dữ liệu đầu vào mở rộng. Số 11 ở đầu được thêm vào để phép tính đơn giản hơn và thuận tiện cho việc tính toán. Khi đó, phương trình (1) có thể được viết lại dưới dạng:

$$y \approx \bar{x}w = \hat{y}$$

1.1 Sai số dự đoán

Chúng ta mong muốn rằng sừ sai khác e giữa giá trị thực y và giá trị dự đoán $\hat{y}$ là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

$$\frac{1}{2}e^2 = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - \bar{x}w)^2$$

Trong đó hệ số $\frac{1}{2}$ (lại) để thuận tiện cho việc tính toán(khi tính đạo hàm thì số $\frac{1}{2}$ sẽ bị triệt tiêu). Chúng ta cần e^2 vì $e = y - \hat{y}$ có thể là một số âm, việc nói e nhỏ nhất sẽ không đúng vì khi $e = -\infty$ là rất nhỏ nhưng sự sai lệch là rất lớn.

1.2 Hàm mất mát

Điều tương tự xảy ra với tất cả các cặp (input, output) $(x_i, y_i), i = 1,2, \dots, N$, với N là số lượng 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:

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \bar{x}_i w)^2 \qquad (2)$$

Hàm số $\mathcal{L}(w)$ được gọi là hàm mất mát (loss function) của bài toán Linear Regression. Chúng ta luôn mong muốn rằng sự mất mát (sai só) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số w sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị

của w làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là điểm tối ưu (optimal point), ký hiệu:

$$\mathrm{w}^* = \arg\min_{\mathrm{w}} \mathcal{L}(\mathrm{w})$$

Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát (2). Đặt $\bar{\mathrm{X}} = [\bar{\mathrm{x}}_1; \bar{\mathrm{x}}_2; \ldots; \bar{\mathrm{x}}_\mathrm{N}]$ là một vector cột chứa tất cả các output của training data; $\bar{\mathrm{X}} = [\bar{\mathrm{x}}_1; \bar{\mathrm{x}}_2; \ldots; \bar{\mathrm{x}}_\mathrm{N}]$ là ma trận dữ liệu đầu vào (mở rộng) mà mỗi hang của nó là một điểm dữ liệu. Khi đó hàm số mất mát $\mathcal{L}(\mathrm{w})$ được viết dưới dạng ma trận đơn giản hơn:

$$\mathcal{L}(\mathrm{w}) = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}} (\mathrm{y}_\mathrm{i} - \bar{\mathrm{x}}_{\mathrm{iw}})^2$$
$$= \frac{1}{2}|\mathrm{y} - \bar{\mathrm{X}}\mathrm{w}|_2^2 \qquad (3)$$

Với $||z||_2$ là Euclidean norm (chuẩn Euclid, hay khoảng cách Euclid), nói cách khác $||z||_2^2$ là tổng bình phương mỗi phần tử của vector z. Tới đây, ta đã có một dạng đơn giản của hàm mất mát được viết như phương trình (3).

2.2 Nghiệm cho bài toán Linear Regression

Cách phổ biến để tìm nghiệm cho một bài toán tối ưu (chúng ta đã biết từ khi học cấp 3) là giải phương trình đạo hàm (gradient) bằng 0! Tất nhiên đó là khi việc tính đạo hàm và việc giải phương trình đạo hàm bằng 0 không quá phức tạp. Thật may mắn, với mô hình tuyến tính, hai việc này là khả thi.

Đạo hàm theo w của hàm mất mát là:

$$\frac{\partial \mathcal{L}(\mathrm{w})}{\partial \mathrm{w}} = \bar{\mathrm{X}}^\mathrm{T}(\bar{\mathrm{X}}\mathrm{w} - \mathrm{y})$$

Phương trình đạo hàm bằng 0 tương đương với:

$$\bar{\mathrm{X}}^\mathrm{T}\bar{\mathrm{X}}\mathrm{w} = \bar{\mathrm{X}}^\mathrm{T}\mathrm{y}\ \mathrm{b}$$

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

$$\mathbf{w} = \mathbf{A}^\dagger\mathbf{b} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^\dagger\bar{\mathbf{X}}^T\mathbf{y} \quad (5)$$

3 Ví dụ trên python

Có nhiều cách khác nhau để thực hiện hồi quy tuyến tính, hoặc sử dụng scikit-learn, stats model, numpy, hoặc scipy.

Trong bài này, tôi sẽ thực hiện hồi quy tuyến tính bằng cách sử dụng numpy cho các phép toán trên ma trận và scikit-learn để kiểm chứng lại nảy giờ tôi viết không phải là nhảm.

3.1 Simple Linear Regression

3.1.1 Bài toán

Giả sử rằng hai biến có quan hệ tuyến tính. Do đó, ta thử tìm một hàm tuyến tính để dự đoán giá trị (y) chính xác nhất có thể như một hàm của biến hoặc biến độc lập(x).

Chúng ta hãy xem xét một tập dữ liệu trong đó chúng ta có một giá trị x là chiều cao và y là cân nặng cần dự đoán như sau:

Chiều cao(cm)	Cân nặng(kg)	Chiều cao(cm)	Cân nặng(kg)
147	49	168	60
150	50	170	72
153	51	173	63
155	52	175	64
158	54	178	66
160	56	180	67
163	58	183	68
165	59		

Bài toán đặt ra là: liệu có thể dự đoán cân nặng của một người dựa vào chiều cao của họ không? (Trên thực tế, tất nhiên là không, vì cân nặng còn phụ thuộc vào nhiều yếu tố khác nữa, thể tích chẳng hạn). Vì blog này nói về các thuật toán Machine Learning đơn giản nên tôi sẽ giả sử rằng chúng ta có thể dự đoán được.

Chúng ta có thể thấy là cân nặng sẽ tỉ lệ thuận với chiều cao (càng cao càng nặng), nên có thể sử dụng Linear Regression model cho việc dự đoán này. Để kiểm tra độ chính xác của model tìm được, chúng ta sẽ giữ lại cột 155 và 160 cm để kiểm thử, các cột còn lại được sử dụng để huấn luyện (train) model.

Ta cần định nghĩa:

X là feature vector , i. e $\mathrm{x} = [\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{n}]$

y là response vector , i. e $\mathrm{y} = [\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_\mathrm{n}]$

Cho n quan sát(trong ví dụ trên, n=15)

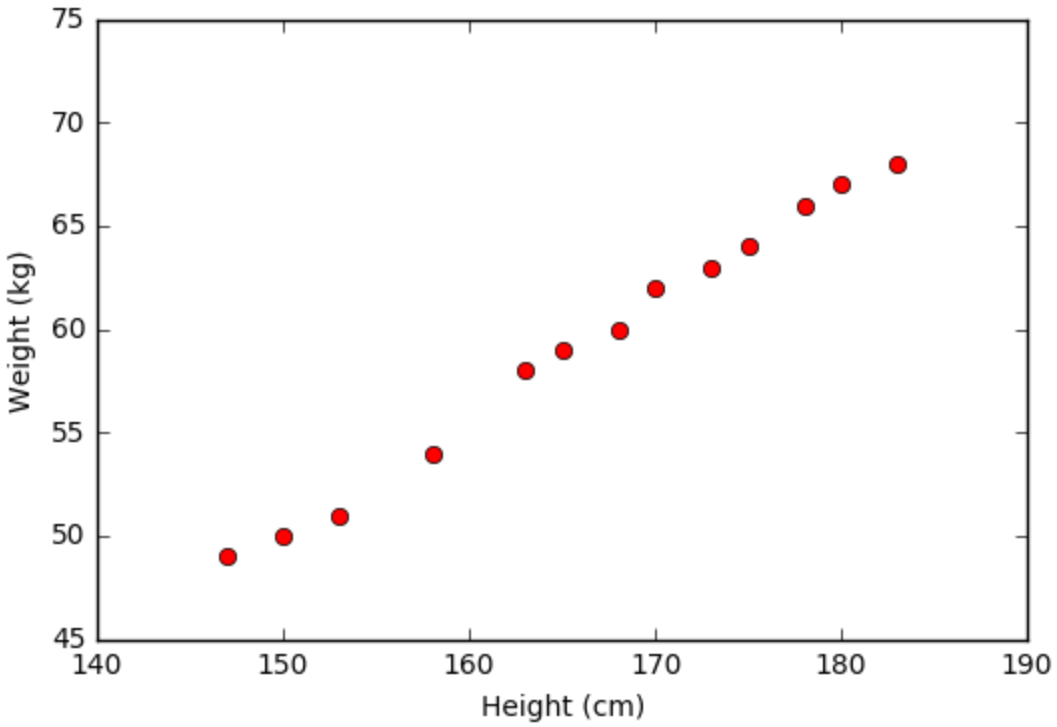
3.1.2 Hiển thị dữ liệu trên đồ thị

Chúng ta cần các thư viện cần thiết. numpy cho đại số tuyến tính và matplotlib cho việc vẽ hình.

```
1. # To support both python 2 and python 3
2. from __future__ import division, print_function, unicode_literals
3. import numpy as np
4. import matplotlib.pyplot as plt
```

Tiếp theo, chúng ta khai báo và biểu diễn dữ liệu trên đồ thị

```
1. # height (cm)
2. X = np.array([[147, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183]]).T

3. # weight (kg)
4. y = np.array([[ 49, 50, 51,  54, 58, 59, 60, 62, 63, 64, 66, 67, 68]]).T
5. # Visualize data
6. plt.plot(X, y, 'ro')
7. plt.axis([140, 190, 45, 75])
8. plt.xlabel('Height (cm)')
9. plt.ylabel('Weight (kg)')
10. plt.show()
```



Từ đồ thị này ta thấy rằng dữ liệu được sắp xếp gần như theo 1 đường thẳng, vậy mô hình Linear Regression nhiều khả năng sẽ cho kết quả tốt:

```
w =  [[-33.73541021]
 [  0.55920496]]
```

(cân nặng) = w_1*(chiều cao) + w_0

3.1.3 Nghiệm của bài toán

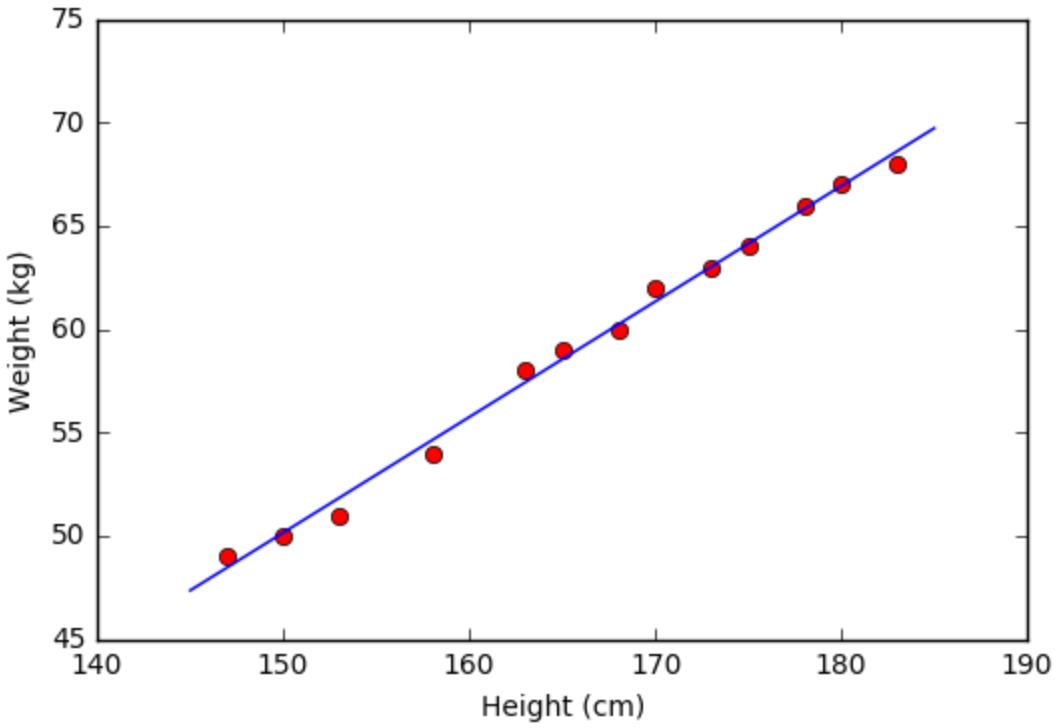
Tiếp theo, chúng ta sẽ tính toán các hệ số w_1 và w_0 dựa vào công thức (5). Chú ý: giả nghịch đảo của một ma trận A trong Python sẽ được tính
bằng numpy.linalg.pinv(A), pinv là từ viết tắt của *pseudo inverse*.

```
# Building Xbar
one = np.ones((X.shape[0], 1))
Xbar = np.concatenate((one, X), axis = 1)

```

```
# Calculating weights of the fitting line
A = np.dot(Xbar.T, Xbar)
b = np.dot(Xbar.T, y)
w = np.dot(np.linalg.pinv(A), b)
print('w = ', w)
# Preparing the fitting line
w_0 = w[0][0]
w_1 = w[1][0]
x0 = np.linspace(145, 185, 2)
y0 = w_0 + w_1*x0

# Drawing the fitting line
plt.plot(X.T, y.T, 'ro')     # data
plt.plot(x0, y0)             # the fitting line
plt.axis([140, 190, 45, 75])
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.show()
```

Kết quả nhận được:

```
w =  [[-33.73541021]
 [  0.55920496]]
```



Từ đồ thị bên trên ta thấy rằng các điểm dữ liệu màu đỏ nằm khá gần đường thẳng dự đoán màu xanh. Vậy mô hình Linear Regression hoạt động tốt với tập dữ liệu *training*. Bây giờ, chúng ta sử dụng mô hình này để dự đoán cân nặng của hai người có chiều cao 155 và 160 cm mà chúng ta đã không dùng khi tính toán nghiệm.

```
1. y1 = w_1*155 + w_0
2. y2 = w_1*160 + w_0
3.
4. print( u'Predict weight of person with height 155 cm: %.2f (kg), real number: 52 (kg)'  %(y1) )
5. print( u'Predict weight of person with height 160 cm: %.2f (kg), real number: 56 (kg)'  %(y2) )
```

Kết quả nhận được:

Predict weight of person with height 155 cm: 52.94 (kg), real number: 52 (kg)
Predict weight of person with height 160 cm: 55.74 (kg), real number: 56 (kg)

Chúng ta thấy rằng kết quả dự đoán khá gần với số liệu thực tế.

3.1.4 Nghiệm theo thư viện scikit-learn

Tiếp theo, chúng ta sẽ sử dụng thư viện scikit-learn của Python để tìm nghiệm.

```
1. from sklearn import datasets, linear_model
2. 
3. # fit the model by Linear Regression
4. regr = linear_model.LinearRegression(fit_intercept=False) # fit_intercept = False f
   or calculating the bias
5. regr.fit(Xbar, y)
6. 
7. # Compare two results
8. print( 'Solution found by scikit-learn  : ', regr.coef_ )
9. print( 'Solution found by (5): ', w.T)
```

Kết quả nhận được:

```
1. Solution found by scikit-learn  :  [[  -33.73541021 0.55920496]]
2. Solution found by (5):  [[  -33.73541021 0.55920496 ]]
```

Chúng ta thấy rằng hai kết quả thu được như nhau! (*Nghĩa là tôi đã không mắc lỗi nào trong cách tìm nghiệm ở phần trên*)

3.2 Multiple linear regression

Multiple linear regression cố gắng mô hình hóa mối quan hệ giữa hai hoặc nhiều tính năng và response bằng cách kết hợp một phương trình tuyến tính với dữ liệu quan sát được.

Rõ ràng, nó chỉ là việc mở rộng hồi quy tuyến tính đơn giản(simple linear regression.)

3.2.1 Bài toán

Cũng là dự đoán giá nhà, nhưng tôi có sử dụng The Boston Housing Dataset gồm giá nhà ở nhiều nơi khác nhau tại Thành Phố Boston. Cùng với giá, tập dữ liệu cũng cũng cung cấp thông tin như Crime(CRIM), khu vực kinh doanh không bán lẻ trong thị trấn(INDUS), tuổi của những người sở hữu ngồi nhà(AGE) và nhiều thuộc tính khác được mô tả tại đây.

3.2.2 Hiển thị dữ liệu

Tập dữ liệu có sẵn tại đây. Tuy nhiên, để đơn giản tôi sẽ sử dụng scikit-learn, có thể import nó ngay từ scikit-learn. (Do thời lượng cũng như độ dài của trang sách, tôi sẽ trình bày một bài riêng về vấn đề đọc datasets trong Machine Learning).

Trước tiên, chúng ta import thư viện cần thiết

```
1. import matplotlib.pyplot as plt
2. import numpy as np
3. from sklearn import datasets, linear_model, metrics
```

Tiếp theo, chúng ta tìm và biểu diễn dữ liệu trên một đồ thị.

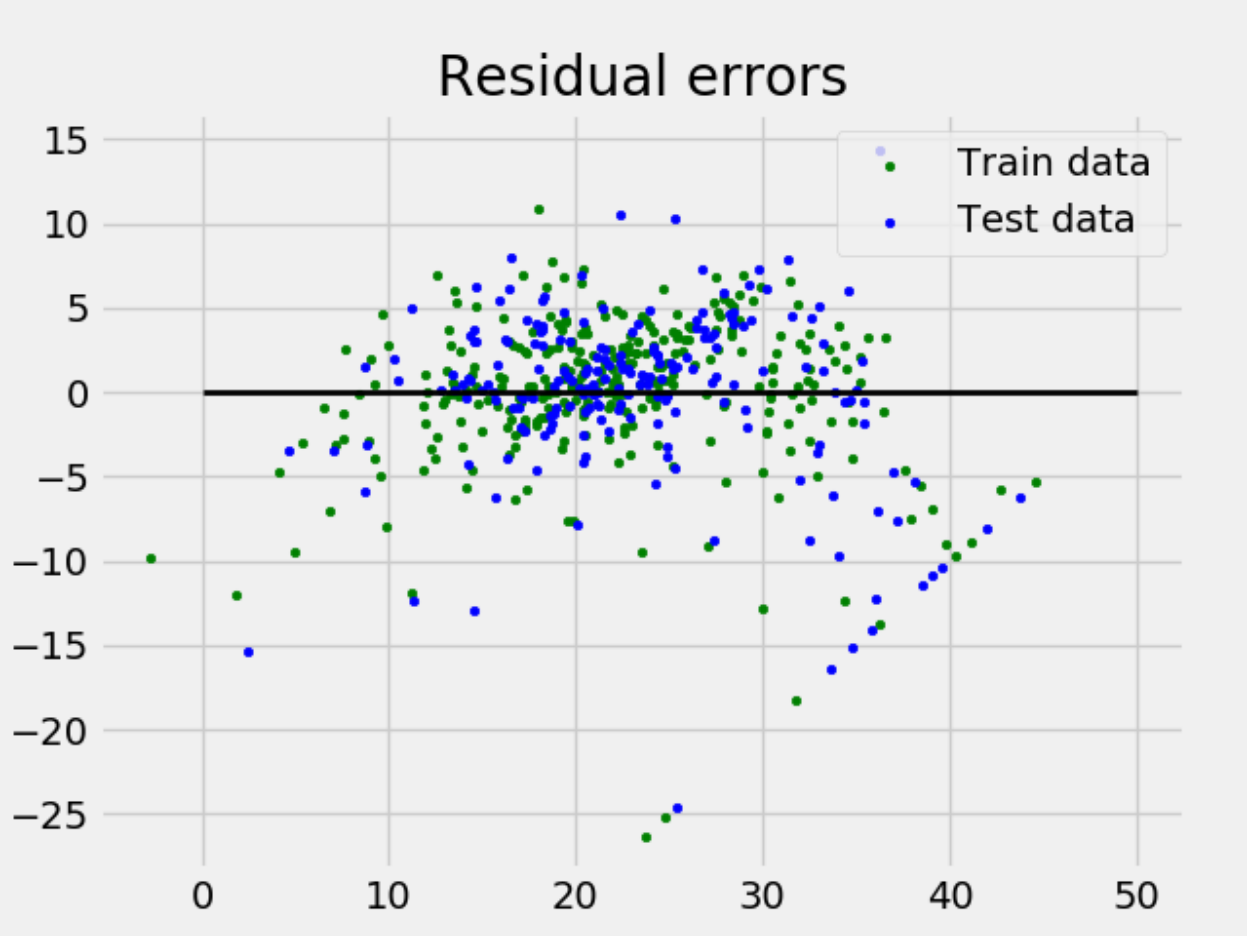
```
1. # load the boston dataset
2. boston = datasets.load_boston(return_X_y=False)
3.
4. # defining feature matrix(X) and response vector(y)
5. X = boston.data
6. y = boston.target
7.
8. # splitting X and y into training and testing sets
9. from sklearn.model_selection import train_test_split
10. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
11.                                                    random_state=1)
12.
13. # create linear regression object
14. reg = linear_model.LinearRegression()
15.
16. # train the model using the training sets
17. reg.fit(X_train, y_train)
18.
19. # regression coefficients
20. print('Coefficients: \n', reg.coef_)
21.
22. # variance score: 1 means perfect prediction
23. print('Variance score: {}'.format(reg.score(X_test, y_test)))
24.
25. # plot for residual error
26.
```

```
## setting plot style
plt.style.use('fivethirtyeight')

## plotting residual errors in training data
plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train,
            color = "green", s = 10, label = 'Train data')

## plotting residual errors in test data
plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test,
            color = "blue", s = 10, label = 'Test data')

## plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2)

## plotting legend
plt.legend(loc = 'upper right')

## plot title
plt.title("Residual errors")

## function to show plot
plt.show()
```

Kết quả nhận được:

```
Coefficients:
[ -8.80740828e-02   6.72507352e-02   5.10280463e-02   2.18879172e+00
-1.72283734e+01   3.62985243e+00   2.13933641e-03  -1.36531300e+00
2.88788067e-01  -1.22618657e-02  -8.36014969e-01   9.53058061e-03
-5.05036163e-01]
Variance score: 0.720898784611
```

Và **Residual Error plot** như sau:



Tron ví dụ trên, chúng ta xác định độ chính xác bằng cách sử dụng Explained Variance Score.

Ta cần xác định:

explained_variance_score = 1 – Var{y – y'}/Var{y}

Trong đó y' là đầu ra ước lượng, y đầu ra cuối cùng tương ứng và Var là phương sai, bình phương độ lệch chuẩn.

4 Thảo luận và hạn chế

Dưới đây là các tình huống cơ bản mà mô hình hồi quy tuyến tính tạo ra liên quan đến tập dữ liệu mà nó được áp dụng.

4.1.1 Mối quan hệ tuyến tính:

Mối quan hệ giữa response và feature phải là tuyền tính. Giả thiết tuyến tính có thể được kiểm tra bằng scatter plots. Như hình dưới đây, hình thứ nhất đại diện cho các biến là tuyến tính và hình 2 và 3 có nhiều khả năng là không tuyến tính. Do đó, hình 1 sẽ đưa ra dự đoán tốt hơn bằng cách sử dụng hồi quy tuyến tính

4.1.2 Ít hoặc không có đa cộng tuyến:

Giả sử rằng có ít hoặc không có đa cộng tuyến trong dữ liệu. Đa cộng tuyến xảy ra khi feature(biến đọc lập) là không độc lập với nhau.

4.1.3 Ít hoặc không có tính tương quan:

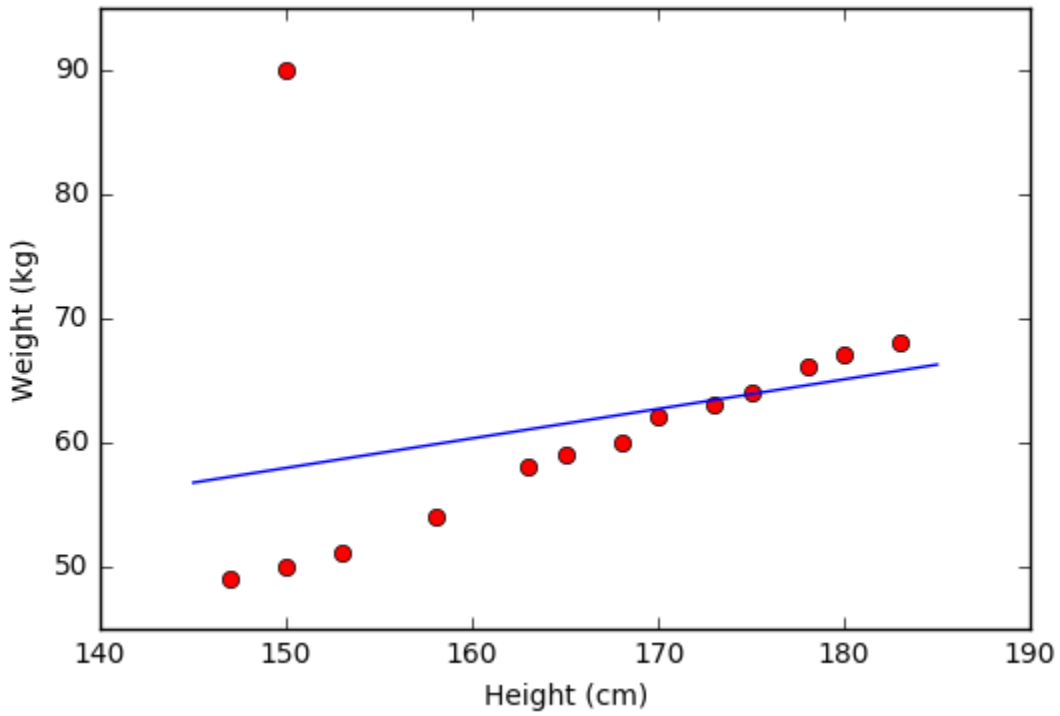
Giả thiết khác là có rất ít hoặc không có sự tương quan trong dự liệu. Tương quan xảy ra khi các lỗi còn lại không phụ liệu vào feature khác.

4.1.4 Homoscedasiticity:

Mô tả tình huống mà trong đó dữ liệu có nhiễu hoặc nhiễu ngẫu nhiên trong mối quan hệ giữa biến độc lập và biến phụ thuộc. Như hình dưới đây, hình 1 có tính đồng nhất trong khi hình 2 không có tính đồng nhất

4.1.5 Hạn chế

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* (150 cm, 90kg) thì kết quả sẽ sai khác đi rất nhiều. Xem hình dưới đây:



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 biểu 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.

5 Ứng dụng

5.1 Đường xu hướng:

Biểu diễn biến thể trong một số dữ liệu so với thời gian(như GDP, giá dầu, …). Những xu hướng này thường theo mối quan hệ tuyến tính. Do đó, hồi quy tuyến tính có thể được áp dụng để dự đoán các giá trị trong tương lai. Tuy nhiên, phương pháp này thiếu khoa học trong trường hợp các thay đổi tiềm năng có thể ảnh hưởng.

5.2 Kinh tế:

Hồi quy tuyến tính là công cụ chủ yếu trong kinh tế học. Ví dụ, nó sử dụng để dự đoán chi tiêu tiêu dùng, chi tiêu đầu tư cố định, đầu tư hàng tồn kho, mua hàng xuất khẩu của một quốc gia, chi tiêu nhập khẩu, nhu cầu lao động và cung lao động.

5.3 Tài chính:

Mô hình tài sản giá vốn sử dụng hồi quy tuyến tính để phân tích và định lượng các rủi ro hệ thống của một khoản đầu tư.

5.4 Sinh học:

Hồi quy tuyến tính được sử dụng để mô hình mối quan hệ nguyên nhân-kết quả giữa các tham số trong hệ thống sinh học.

6 Tài liệu tham khảo

http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html. (n.d.). *Linear Model.*

https://machinelearningmastery.com/simple-linear-regression-tutorial-for-machine-learning/. (n.d.). *Simple Linear Regression Tutorial for Machine Learning.*

https://www.geeksforgeeks.org/linear-regression-python-implementation/. (n.d.). *Linear Regression (Python Implementation).*

NG, A. (n.d.). *Machine Learning on Coursera.*

Vu, T. H. (n.d.). *Machine Learning co ban.*

Wikipedia. (n.d.). *Linear regression.*