« Bài 2: Phân nhóm các thuật toán Machine Learning (/2016/12/27/categories/)
Bài 4: K-means Clustering » (/2017/01/01/kmeans/)

Bài 3: Linear Regression

Linear-models (/tags#Linear-models) Regression (/tags#Regression) Supervised-learning (/tags#Supervised-learning)

Dec 28, 2016

Trong bài này, tôi sẽ giới thiệu một trong những thuật toán cơ bản nhất (và đơn giản nhất) của Machine Learning. Đây là một thuật toán *Supervised learning* có tên **Linear Regression** (Hồi Quy Tuyến Tính). Bài toán này đôi khi được gọi là *Linear Fitting* (trong thống kê) hoặc *Linear Least Square*. **Trong trang này:**

1. Giới thiệu

Quay lại ví dụ đơn giản được nêu trong bài trước (/2016/12/27/categories/#regression-hoi-quy): một căn nhà rộng x_1 m^2, có x_2 phòng ngủ và cách trung tâm thành phố x_3 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 $y = f(\mathbf{x})$ sẽ có dạng như thế nào. Ở đây $\mathbf{x} = [x_1, x_2, x_3]$ là một vector hàng chứa thông tin *input*, y là một số vô hướng (scalar) biểu diễn *output* (tức giá của căn nhà trong ví dụ này).

Lưu ý về ký hiệu toán học: *trong các bài viết của tôi, các số vô hướng được biểu diễn bởi các chữ cái viết ở dạng không in đậm, có thể viết hoa, ví dụ x_1, N, y, k. Các vector được biểu diễn bằng các chữ cái thường in đậm, ví dụ $\mathbf{y}, \mathbf{x}_1$. Các ma trận được biểu diễn bởi các chữ viết hoa in đậm, ví dụ $\mathbf{X}, \mathbf{Y}, \mathbf{W}$.*

Một cách đơn giản nhất, chúng ta có thể thấy rằng: i) diện tích nhà càng lớn thì giá nhà càng cao; ii) số lượng phòng ngủ càng lớn thì giá nhà càng cao; iii) 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:

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

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + w_0 \quad (1)$$

trong đó, w_1, w_2, w_3, w_0 là các hằng số, w_0 còn được gọi là bias. Mối quan hệ $y \approx f(\mathbf{x})$ 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 $\{w_1, w_2, w_3, w_0\}$ chính vì vậy được gọi là bài toán Linear Regression.

Chú ý 1: y là giá trị thực của *outcome* (dựa trên số liệu thống kê chúng ta có trong tập *training data*), trong khi $\hat{y}$ là giá trị mà mô hình Linear Regression dự đoán được. Nhìn chung, y và $\hat{y}$ là hai giá trị khác nhau do có sai số mô hình, tuy nhiên, chúng ta mong muốn rằng sự khác nhau này rất nhỏ.

Chú ý 2: *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. Chúng ta sẽ được thấy trong các bài viết sau, *tuyến tính* rất quan trọng và hữu ích trong các bài toán Machine Learning. Kinh nghiệm cá nhân tôi cho thấy, trước khi hiểu được các thuật toán *phi tuyến* (non-linear, không phẳng), chúng ta cần nắm vững các kỹ thuật cho các mô hình *tuyến tính*.

2. Phân tích toán học

2.1. Dạng của Linear Regression

Trong phương trình (1) phía trên, nếu chúng ta đặt $\mathbf{w} = [w_0, w_1, w_2, w_3]^T =$ là vector (cột) hệ số cần phải tối ưu và $\bar{\mathbf{x}} = [1, x_1, x_2, x_3]$ (đọc là *x bar* trong tiếng Anh) là vector (hàng) dữ liệu đầu vào *mở rộng*. Số 1 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:

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

Chú ý rằng $\bar{\mathbf{x}}$ là một vector hàng. (Xem thêm về ký hiệu vector hàng và cột tại đây (/math/#luu-y-ve-ky-hieu))

2.2. Sai số dự đoán

Chúng ta mong muốn rằng sự sai khác e giữa giá trị thực y và giá trị dự đoán $\hat{y}$ (đọc là *y hat* trong tiếng Anh) là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

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

trong đó hệ số $\frac{1}{2}$ (*lại*) là 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. Bạn đọc có thể tự đặt câu hỏi: **tại sao không dùng trị tuyệt đối $|e|$ mà lại dùng bình phương e^2 ở đây?** Câu trả lời sẽ có ở phần sau.

2.3. Hàm mất mát

Điều tương tự xảy ra với tất cả các cặp *(input, outcome)* $(\mathbf{x}_i, y_i), i = 1, 2, \ldots, N$, 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 $\mathbf{w}$ để hàm số sau đạt giá trị nhỏ nhất:

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

Hàm số $\mathcal{L}(\mathbf{w})$ được gọi là **hàm mất mát** (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 $\mathbf{w}$ sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của $\mathbf{w}$ làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là *điểm tối ưu* (optimal point), ký hiệu:

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

Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát (2). Đặt $\mathbf{y} = [y_1; y_2; \ldots; y_N]$ là một vector cột chứa tất cả các *output* của *training data*; $\bar{\mathbf{X}} = [\bar{\mathbf{x}}_1; \bar{\mathbf{x}}_2; \ldots; \bar{\mathbf{x}}_N]$ 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 $\mathcal{L}(\mathbf{w})$ được viết dưới dạng ma trận đơn giản hơn:

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

$$= \frac{1}{2}\|\mathbf{y} - \bar{\mathbf{X}}\mathbf{w}\|_2^2 \quad (3)$$

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

2.4. Nghiệm cho bài toán Linear Regression

Cách phổ biến nhất 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 các mô hình tuyến tính, hai việc này là khả thi.

Đạo hàm theo $\mathbf{w}$ của hàm mất mát là:

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

Các bạn có thể tham khảo bảng đạo hàm theo vector hoặc ma trận của một hàm số trong mục D.2 của tài liệu này (https://ccrma.stanford.edu/~dattorro/matrixcalc.pdf). *Đến đây tôi xin quay lại câu hỏi ở phần Sai số dự đoán phía trên về việc tại sao không dùng trị tuyệt đối mà lại dùng bình phương. Câu trả lời là hàm bình phương có đạo hàm tại mọi nơi, trong khi hàm trị tuyệt đối thì không (đạo hàm không xác định tại 0)*.

Phương trình đạo hàm bằng 0 tương đương với:

$$\bar{\mathbf{X}}^T\bar{\mathbf{X}}\mathbf{w} = \bar{\mathbf{X}}^T\mathbf{y} \triangleq \mathbf{b} \quad (4)$$

(ký hiệu $\bar{\mathbf{X}}^T\mathbf{y} \triangleq \mathbf{b}$ nghĩa là *đặt* $\bar{\mathbf{X}}^T\mathbf{y}$ *bằng* $\mathbf{b}$).

Nếu ma trận vuông $\mathbf{A} \triangleq \bar{\mathbf{X}}^T\bar{\mathbf{X}}$ khả nghịch (non-singular hay invertible) thì phương trình (4) có nghiệm duy nhất: $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$.

Vậy nếu ma trận $\mathbf{A}$ không khả nghịch (có định thức bằng 0) thì sao? Nếu các bạn vẫn nhớ các kiến thức về hệ phương trình tuyến tính, trong trường hợp này thì hoặc phương trinh (4) vô nghiệm, hoặc là nó có vô số nghiệm. Khi đó, chúng ta sử dụng khái niệm *giả nghịch đảo* (https://vi.wikipedia.org/wiki/Giả_nghịch_đảo_Moore–Penrose) $\mathbf{A}^\dagger$ (đọc là *A dagger* trong tiếng Anh). (*Giả nghịch đảo (pseudo inverse) là trường hợp tổng quát của nghịch đảo khi ma trận không khả nghịch hoặc thậm chí không vuông. Trong khuôn khổ bài viết này, tôi xin phép được lược bỏ phần này, nếu các bạn thực sự quan tâm, tôi sẽ viết một bài khác chỉ nói về giả nghịch đảo. Xem thêm: Least Squares, Pseudo-Inverses, PCA & SVD (http://www.sci.utah.edu/~gerig/CS6640-F2012/Materials/pseudoinverse-cis61009sl10.pdf).*)

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

3.1. Bài toán

Trong phần này, tôi sẽ chọn một ví dụ đơn giản về việc giải bài toán Linear Regression trong Python. Tôi cũng sẽ so sánh nghiệm của bài toán khi giải theo phương trình (5) và nghiệm tìm được khi dùng thư viện scikit-learn (http://scikit-learn.org/stable/) của Python. (*Đây là thư viện Machine Learning được sử dụng rộng rãi trong Python*). Trong ví dụ này, dữ liệu đầu vào chỉ có 1 giá trị (1 chiều) để thuận tiện cho việc minh hoạ trong mặt phẳng.

Chúng ta có 1 bảng dữ liệu về chiều cao và cân nặng của 15 người như dưới đây:

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.

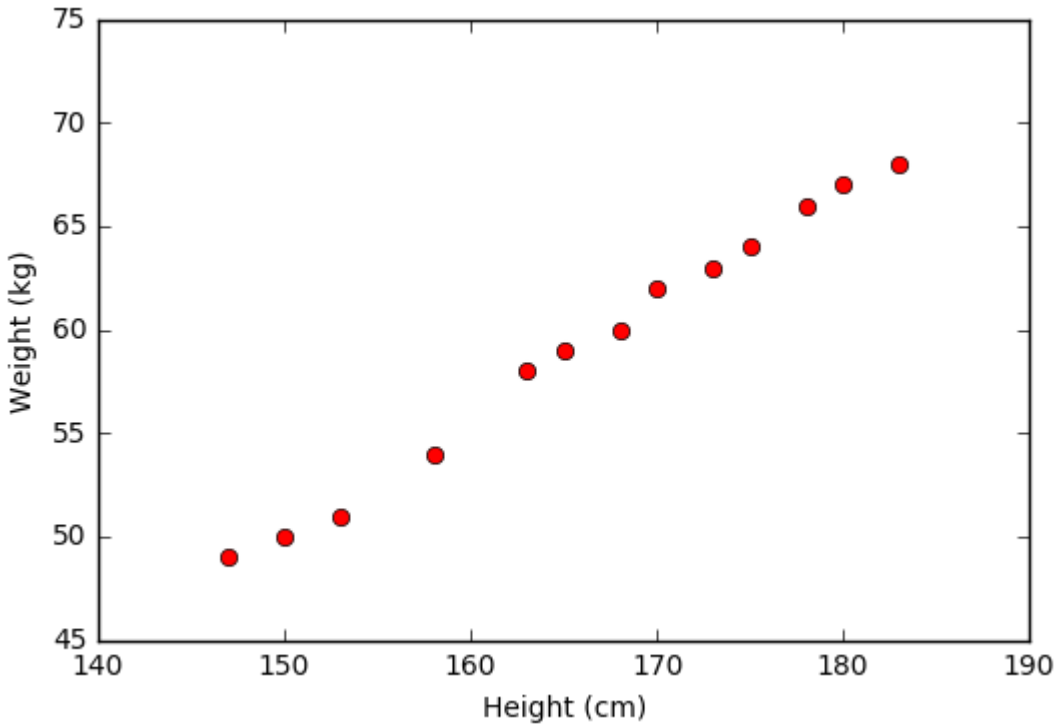
3.2. Hiển thị dữ liệu trên đồ thị

Trước tiên, chúng ta cần có hai thư viện numpy (http://www.numpy.org/) cho đại số tuyến tính và matplotlib (https://matplotlib.org/) cho việc vẽ hình.

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

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

```
# height (cm)
X = np.array([[147, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183]]).T
# weight (kg)
y = np.array([[ 49, 50, 51,  54, 58, 59, 60, 62, 63, 64, 66, 67, 68]]).T
# Visualize data
plt.plot(X, y, 'ro')
plt.axis([140, 190, 45, 75])
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
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:

(cân nặng) = `w_1`*(chiều cao) + `w_0`

3.3. Nghiệm theo công thức

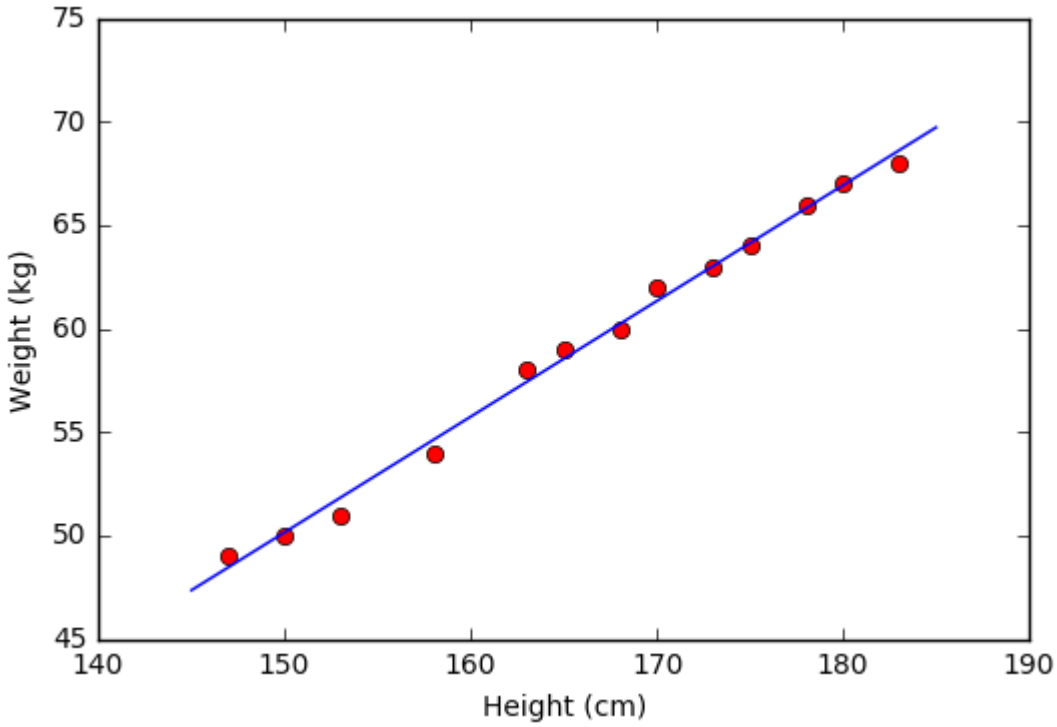
Tiếp theo, chúng ta sẽ tính toán các hệ số `w_1` và `w_0` dựa vào công thức (5). Chú ý: giả nghịch đảo của một ma trận `A` trong Python sẽ được tính bằng `numpy.linalg.pinv(A)`, `pinv` là từ viết tắt của *pseudo inverse*.

```
# 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()
```

```
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.

```
y1 = w_1*155 + w_0
y2 = w_1*160 + w_0

print( u'Predict weight of person with height 155 cm: %.2f (kg), real number: 52 (kg)'  %(y1) )
print( u'Predict weight of person with height 160 cm: %.2f (kg), real number: 56 (kg)'  %(y2) )
```

```
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.

3.4. Nghiệm theo thư viện scikit-learn

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

```
from sklearn import datasets, linear_model

# fit the model by Linear Regression
regr = linear_model.LinearRegression(fit_intercept=False) # fit_intercept = False for calculating
regr.fit(Xbar, y)

# Compare two results
print( 'Solution found by scikit-learn  : ', regr.coef_ )
print( 'Solution found by (5): ', w.T)
```

```
    Solution found by scikit-learn  :  [[  -33.73541021 0.55920496]]
    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*)

Source code Jupyter Notebook cho bài này. (https://github.com/tiepvupsu/tiepvupsu.github.io/blob/master/assets/LR/LR.ipynb)

4. Thảo luận

4.1. Các bài toán có thể giải bằng Linear Regression

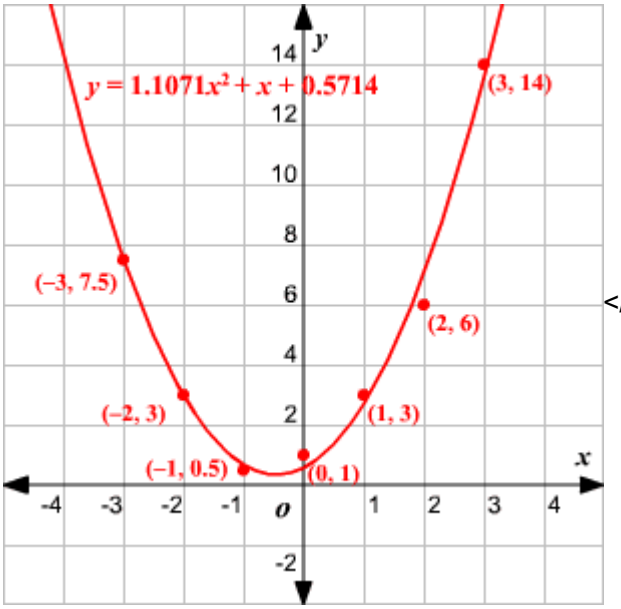
Hàm số $y \approx f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ là một hàm tuyến tính theo cả $\mathbf{w}$ và $\mathbf{x}$. Trên thực tế, Linear Regression có thể áp dụng cho các mô hình chỉ cần tuyến tính theo $\mathbf{w}$. Ví dụ:

$$y \approx w_1x_1 + w_2x_2 + w_3x_1^2 +$$

$$+w_4\sin(x_2) + w_5x_1x_2 + w_0$$

là một hàm tuyến tính theo $\mathbf{w}$ và vì vậy cũng có thể được giải bằng Linear Regression. Với mỗi dữ liệu đầu vào $\mathbf{x} = [x_1; x_2]$, chúng ta tính toán dữ liệu mới $\tilde{\mathbf{x}} = [x_1, x_2, x_1^2, \sin(x_2), x_1x_2]$ (đọc là *x tilde* trong tiếng Anh) rồi áp dụng Linear Regression với dữ liệu mới này.

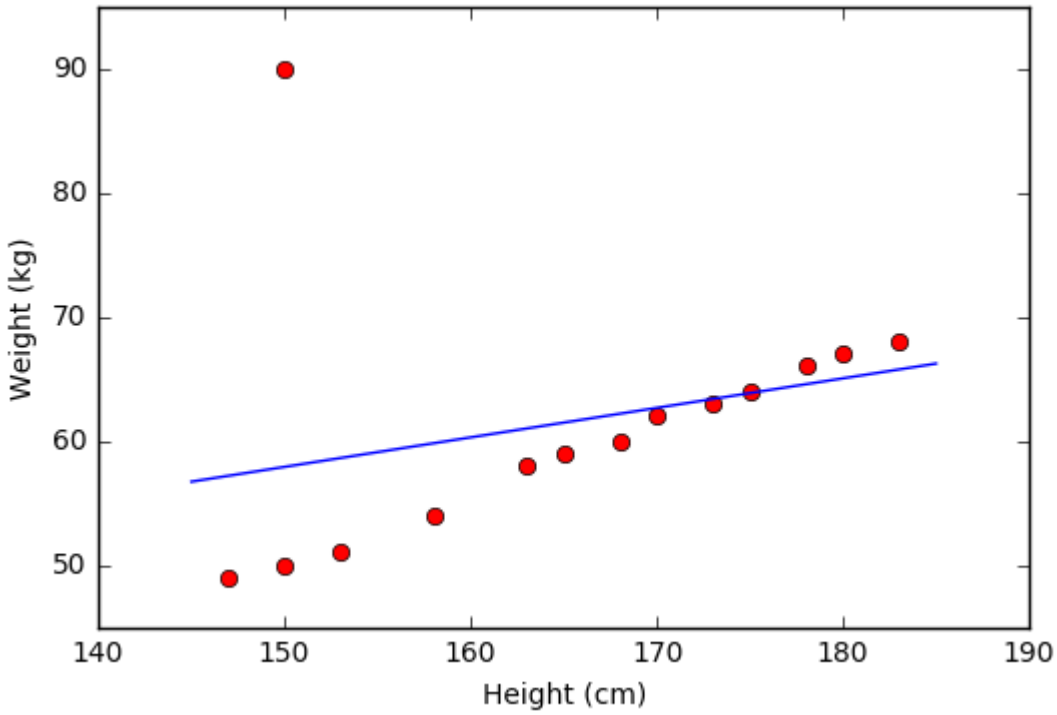
Xem thêm ví dụ về Quadratic Regression (http://www.varsitytutors.com/hotmath/hotmath_help/topics/quadratic-regression) (Hồi Quy Bậc Hai).



Quadratic Regression (Nguồn: Quadratic Regression (http://www.varsitytutors.com/hotmath/hotmath_help/topics/quadratic-regression))

4.2. 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* (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. Hơn nữa, chúng ta sẽ tự hỏi: làm thế nào để xác định được các hàm $x_1^2, \sin(x_2), x_1x_2$ như ở trên?!

4.3. Các phương pháp tối ưu

Linear Regression là một mô hình đơn giản, lời giải cho phương trình đạo hàm bằng 0 cũng khá đơn giản. *Trong hầu hết các trường hợp, chúng ta không thể giải được phương trình đạo hàm bằng 0.*

Nhưng có một điều chúng ta nên nhớ, **còn tính được đạo hàm là còn có hy vọng**.

5. Tài liệu tham khảo

1. Linear Regression - Wikipedia (https://en.wikipedia.org/wiki/Linear_regression)
2. Simple Linear Regression Tutorial for Machine Learning (http://machinelearningmastery.com/simple-linear-regression-tutorial-for-machine-learning/)
3. Least Squares, Pseudo-Inverses, PCA & SVD (http://www.sci.utah.edu/~gerig/CS6640-F2012/Materials/pseudoinverse-cis61009sl10.pdf)