Máy học
Chương 2: Linear Regression

Khoa CNTT, Đại học Kỹ thuật - Công nghệ Cần Thơ
Lưu hành nội bộ

Nội dung

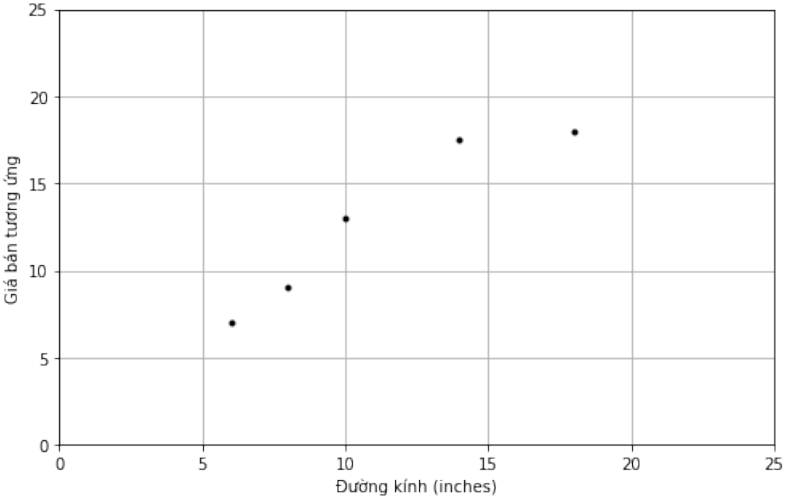
Linear regression qua một ví dụ

- Giả sử chúng ta muốn biết giá mua của 1 bánh pizza.
- Dự đoán giá pizza dựa trên thuộc tính của nó.
- Xây dựng mô hình quan hệ giữa kích thước và giá tiền.

Training data

Training instance	Đường kính (inches)	Giá ($)
1	6	7
2	8	9
3	10	13
4	14	17.5
5	18	18

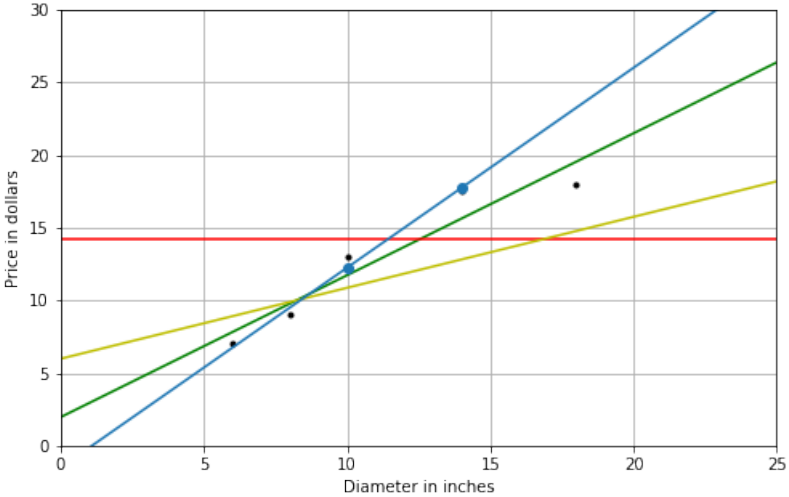
```
import matplotlib.pyplot as plt

X = [[6], [8], [10], [14],   [18]]
y = [[7], [9], [13], [17.5], [18]]

plt.figure(figsize=(8,5))
plt.xlabel('Duong kinh (inches)')
plt.ylabel('Gia ban tuong ung')
plt.plot(X, y, 'k.')
plt.axis([0, 25, 0, 25])
plt.grid(True)
plt.show()
```



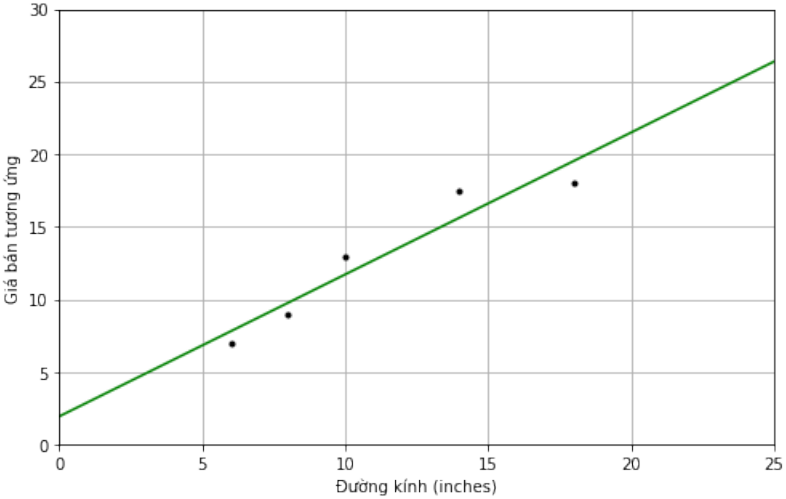
Pizza có đường kính 12 và 20 inch giá bao nhiêu?

Linear regression - hồi quy tuyến tính

- Mối quan hệ tuyến tính giữa giá bán và đường kính
  - Tuyến tính dương (positive): cùng tăng/giảm
  - Tuyến tính âm (negative):khác tăng/giảm
- Có đường liên kết giữa các điểm dữ liệu sao cho sai số dự đoán là nhỏ nhất


Price in dollars
Diameter in inches
0
5
10
15
20
25
30

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

X = [[6], [8], [10], [14],   [18]]
y = [[7], [9], [13], [17.5], [18]]

plt.figure(figsize=(8,5))
plt.xlabel('Duong kinh (inches)')
plt.ylabel('Gia ban tuong ung')
plt.axis([0, 25, 0, 30])
plt.grid(True)

X2 = [[0], [10], [14], [25]]
model = LinearRegression()
model.fit(X, y)
y2 = model.predict(X2)
plt.plot(X, y, 'k.')
plt.plot(X2, y2, 'g-')
plt.show()
```


Giá bán tương ứng
Đường kính (inches)
0
5
10
15
20
25
30

Residuals error

- Trước khi học (training), mỗi giá trị x ta có 1 giá trị y tương tương
- Sau khi học, mỗi giá trị x có 1 giá trị $\hat{y}$ dự đoán tương ứng.
- Sai biệt giữa y và $\hat{y}$ của tất cả n điểm dữ liệu được tính theo công thức sau:

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1}$$

- Được gọi chung là hàm chi phí (cost function)

Residuals error

```
# Code manually
sum = 0.0
for i in range(len(y)):
    sum += (model.predict(X[i][0]) - y[i])**2
print("residual sum of squares: " + str(sum[0][0]))

#residual sum of squares: 28.5

# Use numpy library
import numpy as np
print("residual sum of squares: " + str(np.sum(model.predict(X) - y)
    **2))

#residual sum of squares: 28.5
```

$\hat{y}$

```
33 print(model.predict(6))
34 print(model.predict(8))
35 print(model.predict(10))
36 print(model.predict(14))
37 print(model.predict(18))
38
39 #[[ 6.75]]
40 #[[ 9.5]]
41 #[[ 12.25]]
42 #[[ 17.75]]
43 #[[ 23.25]]
```

$\hat{y}$

- Bài toán tuyến tính hồi quy đơn giản, ta có giá bánh pizza chỉ phụ thuộc vào 1 biến đường kính
- Phương trình đường tuyến tính có dạng đơn giản sau:

$$\hat{y} = \alpha + \beta x \tag{2}$$

- Mục tiêu là tìm giá trị của α và β sao cho cost function nhỏ nhất.
- Giải công thức trên, chúng ta cần tính sự khác biệt (variance) của x và tính hiệp biến (covariance) của x và y.

Variance

- Variance là phép đo độ lệt giá trị; nếu tất cả các giá trị bằng nhau thì variance = 0
- Variance nhỏ biểu hiện cho các giá trị dao động gần vị trí trung bình.
- Variance được tính bởi công thức:

$$var(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \tag{3}$$

Trong đó, $\bar{x}$ là trung bình của các giá trị x.

Variance

```
44 xbar = (6 + 8 + 10 + 14 + 18) / 5
45 variance = ((6 - xbar)**2 + (8 - xbar)**2 + (10 - xbar)**2 + (14 - xbar
       )**2 + (18 - xbar)**2) / 4
46 print (variance)
47
48 import numpy as np
49 print(np.var([6, 8, 10, 14, 18], ddof=1))
50
51 #23.2
52 #23.2
```

Covariance

- Covariance thể hiện mối quan hệ giữa 2 biến với nhau.
- Nếu giá trị 1 biến tăng kéo theo giá trị biến còn lại tăng thì ta có covariance dương.
- Nếu giá trị 1 biến tăng kéo theo giá trị biến còn lại giảm thì ta có covariance âm.
- Covariance $= 0$ khi không có sự quan hệ tương quan nào giữa 2 giá trị biến.

Covariance

- Covariance được tính bởi công thức:

$$cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n - 1} \tag{4}$$

Trong đó, $\bar{x}$ là trung bình của các giá trị x, $\bar{y}$ là trung bình của các giá trị y.

Covariance

```
53 xbar = (6 + 8 + 10 + 14 + 18) / 5
54 ybar = (7 + 9 + 13 + 17.5 + 18) / 5
55 cov = ((6 - xbar) * (7 - ybar) + (8 - xbar) * (9 - ybar) + (10 - xbar)
       * (13 - ybar) +
56 (14 - xbar) * (17.5 - ybar) + (18 - xbar) * (18 - ybar)) / 4
57 print (cov)
58
59 import numpy as np
60 print (np.cov([6, 8, 10, 14, 18], [7, 9, 13, 17.5, 18])[0][1])
61
62 #22.65
63 #22.65
```

$\hat{y}$

$$\beta = \frac{cov(x, y)}{var(x)} \tag{5}$$

$$\beta = \frac{22.65}{23.2} = 0.976293$$

$$\alpha = \bar{y} - \beta\bar{x} \tag{6}$$

$$\alpha = 12.9 - 0.976293 * 11.2 = 1.965518$$

Đánh giá mô hình - evaluating the model

- Có nhiều đánh giá mô hình dựa trên các cost function, ví dụ residual error, root mean square error, mean square error
- Chọn mô hình nào có cost function nhỏ nhất.
- Hướng đánh giá khác dựa trên khả năng dự đoán của mô hình sử dụng r-squared.
  - r-squared $= 1$: các giá trị biến được dự đoán chính xác nếu sử dụng model.
  - r-squared $= 0.5$: 50% các giá trị biến được dự đoán chính xác nếu sử dụng model.
  - r-squared $= 0$: không giá trị biến được dự đoán chính xác nếu sử dụng model.

r-squared

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{7}$$

Trong đó:

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{8}$$

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{9}$$

r-squared

```
from sklearn.linear_model import LinearRegression

X = [[6], [8], [10], [14],   [18]]
y = [[7], [9], [13], [17.5], [18]]
X_test = [[8],  [9],   [11], [16], [12]]
y_test = [[11], [8.5], [15], [18], [11]]
model = LinearRegression()
model.fit(X, y)
print ('R-squared: %.4f' % model.score(X_test, y_test))

#R-squared: 0.6620
```

Multiple linear regression

- Giá bánh pizza phụ thuộc vào nhiều yếu tố ví dụ như đồ dày, phần phủ bề mặt
- Công thức hàm tuyến tính trở thành:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{10}$$

- Tổng hợp các giá trị y, ta có công thức thu gọn như sau:

$$y = X\beta \tag{11}$$

Multiple linear regression - training data

Training example	Diameter (inches)	# toppings	Price ($)
1	6	2	7
2	8	1	9
3	10	0	13
4	14	2	17.5
5	18	0	18

Multiple linear regression - test data

Training example	Diameter (inches)	# toppings	Price ($)
1	8	2	11
2	9	0	8.5
3	11	2	15
4	16	2	18
5	12	0	11

Phép toán ma trận

- Ma trận, phép nhân ma trận, phép nghịch đảo ma trận (inversion), phép hoán vị ma trận (transpose)

```
from numpy.linalg import inv
from numpy import dot, transpose

X = [[1,2,3], [4,5,0]]
Y = [[1,2],[0,1],[1,2]]

print(X)
print(Y)

#[[1, 2, 3], [4, 5, 0]]
#[[1, 2], [0, 1], [1, 2]]
```

Phép toán ma trận

- Phép nhân:

```
print(dot(X,Y))
```

- Phép hóa vị:

```
print(transpose(X))
```

- Phép đảo:

```
print(dot(transpose(X),X))

print(inv(dot(transpose(X),X)))
```

Phép toán ma trận

```
print(dot(X,Y))
#[[ 4 10]
#[ 4 13]]
print(transpose(X))
#[[1 4]
#[2 5]
#[3 0]]
print(dot(transpose(X),X))
#[[17 22  3]
#[22 29  6]
#[ 3  6  9]]
print(inv(dot(transpose(X),X)))
#[[ -1.96429801e+15   1.57143841e+15  -3.92859603e+14]
#[  1.57143841e+15  -1.25715073e+15   3.14287682e+14]
#[ -3.92859603e+14   3.14287682e+14  -7.85719205e+13]]
```

Multiple linear regression

$$\beta = (X^T X)^{-1} X^T Y \tag{12}$$

```
from numpy.linalg import inv
from numpy import dot, transpose
X = [[1, 6, 2], [1, 8, 1], [1, 10, 0], [1, 14, 2], [1, 18, 0]]
y = [[7], [9], [13], [17.5], [18]]
print(dot(inv(dot(transpose(X), X)), dot(transpose(X), y)))

#[[ 1.1875    ]
#[ 1.01041667]
#[ 0.39583333]]
```

Multiple linear regression - sklearn

```
from sklearn.linear_model import LinearRegression

X = [[6, 2], [8, 1], [10, 0], [14, 2], [18, 0]]
y = [[7], [9], [13], [17.5], [18]]

model = LinearRegression()
model.fit(X, y)

X_test = [[8, 2], [9, 0], [11, 2], [16, 2], [12, 0]]
y_test = [[11], [8.5], [15], [18], [11]]

predictions = model.predict(X_test)
for i, prediction in enumerate(predictions):
    print ('Predicted: %s, Target: %s' % (prediction, y_test[i]))
print ('R-squared: %.2f' % model.score(X_test, y_test))
```

Multiple linear regression - sklearn

```
#Predicted: [ 10.0625], Target: [11]
#Predicted: [ 10.28125], Target: [8.5]
#Predicted: [ 13.09375], Target: [15]
#Predicted: [ 18.14583333], Target: [18]
#Predicted: [ 13.3125], Target: [11]
#R-squared: 0.77
```