3rd-week-1

Linear Regression

Wang Jianfang
王建芳(in Chinese)

Agenda

- Concept
- Example
- Code

The least square method

- Because of using overall, unknown sample value estimation which for each x_i.

$$b = \hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$a = \widehat{\alpha} = \bar{y} - b\bar{x}$$

- through $y\_hat = a + bx_i$ predict the corresponding value of y.

The derivation of the least square method

- S1. to minimus the error.
- S2.The way we minimize the error is by minimizing the sum of squares of the error. (The reason to limit the error by minimizing the sum of squared errors is to avoid the influence of negative Numbers on the calculation).

The derivation of the least square method

- There is a set of data [x1,y1][x2,y2]... [xn, yn]
- S1. Linear equation set fitting for: $y = a + bx$
- S2. Error: $d_i = y_i - (a + bx_i)$
- S3. $D_i = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2$

Taking the partial derivative of this equation yields the following two equations.

$$\frac{\partial D_i}{\partial a} = \sum_{i=1}^{n} 2(y_i - a - bx_i)(-1) = -2\sum_{i=1}^{n}(y_i - a - bx_i) = -2(\sum_{i=1}^{n} y_i - na - b\sum_{i=1}^{n} x_i) = 0$$

$$\frac{\partial D_i}{\partial b} = \sum_{i=1}^{n} 2(y_i - a - bx_i)(-x_i) = -2\sum_{i=1}^{n}(y_i - a - bx_i)x_i = -2(\sum_{i=1}^{n} x_i y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2) = 0$$

These two equations form a system of equations and solve for A and B.

After sorting out the equation:

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$b = \frac{1}{n}\sum_{i=1}^{n} y_i - \frac{b}{n}\sum_{i=1}^{n} x_i$$

practice

- x = np.array([1,3,2,1,3])
- y = np.array([14,24,18,17,27])
- y=a + bx
- find the a and b by manual.

The least square method

```
import numpy as np
from matplotlib import pylab as pl

#Defining training data
x = np.array([1,3,2,1,3])
y = np.array([14,24,18,17,27])
# The regression equation takes the function
def fit(x,y):
    if len(x) != len(y):
        return
    numerator = 0.0
    denominator = 0.0
    x_mean = np.mean(x)
    y_mean = np.mean(y)
    for i in range(len(x)):
        numerator += (x[i]-x_mean)*(y[i]-y_mean)
        denominator += np.square((x[i]-x_mean))
    print('numerator:',numerator,'denominator:',denominator)
    b0 = numerator/denominator
    b1 = y_mean - b0*x_mean
    return b0,b1
# Define prediction function
def predit(x,b0,b1):
    return b0*x + b1
# Find the regression equation
b0,b1 = fit(x,y)
print('Line is:y = %2.0fx + %2.0f'%(b0,b1))
# prediction
x_test = np.array([0.5,1.5,2.5,3,4])
y_test = np.zeros((1,len(x_test)))
for i in range(len(x_test)):
    y_test[0][i] = predit(x_test[i],b0,b1)
# Drawing figure
xx = np.linspace(0, 5)
yy = b0*xx + b1
pl.plot(xx,yy,'k-')
pl.scatter(x,y,cmap=pl.cm.Paired)
pl.scatter(x_test,y_test[0],cmap=pl.cm.Paired)
pl.show()
```

the least square method

- Created on Sun Apr 19 09:28:18 2020
- Let's use the least square method to find the equation y is equal to ax plus b, which is a and b
- @author: Jeff King
- """
- ##Draw a figure
- import numpy as np
- import matplotlib.pyplot as plt

- x= np.array([1.,2.,3.,4.,5.])
- y= np.array([1.,3.,2.,3.,5.])

- plt.scatter(x,y)
- plt.axis([0,6,0,6])
- plt.show()
- ##Linear regression is obtained by least square method
- x_mean = np.mean(x)
- y_mean = np.mean(y)

- num=0.0
- d=0.0
- for x_i,y_i in zip(x,y):
-     num += (x_i - x_mean)*(y_i - y_mean)
-     d += (x_i - x_mean)**2

- a = num /d
- b = y_mean - a*x_mean

- y_hat = a*x + b

- ##draw the linear equation

- plt.scatter(x,y)
- plt.plot(x,y_hat, color= 'r')
- plt.axis([0,6,0,6])
- plt.show()

homework

- x = [45, 73, 89, 120, 140, 163]
- y= [80, 150, 198, 230, 280, 360]
- $y=ax+b$
- Drawing the figure by pyton.

Questions and Comments?

Thank you!!