Regression and Multivariate Analysis

Prasanta K. Jana,IEEE Senior Member

Department of Computer Science and Engineering
Indian Institute of Technology (ISM), Dhanbad
E-mail: prasantajana@iitism.ac.in

Regression

- A predictive modeling technique where the target variable to be estimated is continuous.
- Applications
  - Predicting a stock market.
  - Forecasting amount of precipitation in a region.
  - Projecting total sale of a company etc.

Preliminaries

Let D be a data set that contains N observations:
i.e., D $= \{(\mathbf{x}_i, y_i) | i = 1, 2, ..., N\}$
$\mathbf{x}_i$ = Set of attributes of ith observations(**explanatory variables**)
y_i = Target (**response variable**)

- Regression is the task of learning a target function f that maps each attribute set $\mathbf{x}$ into a continuous valued output y.

Preliminaries

- The goal of regression is to find a target function that can fit the input data with minimum error
- The error function for a regression task can be expressed as:

  Absolute Error $= \sum_i |y_i - f(\mathbf{x}_i)|$

  Squared Error $= \sum_i (y_i - f(\mathbf{x}_i))^2$

- The data corresponds to measurements of heat flux and skin temperature of a person during sleep
- Predict the skin temperature of a person based on the heat flux measurements generated by a heat sensor

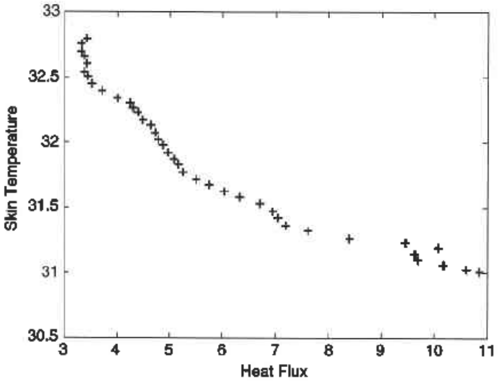
Heat Flux	Skin Temperature
10.858	31.002
10.617	31.021
10.183	31.058
9.7003	31.095
9.852	31.133
10.088	31.188
9.459	31.226
8.3972	31.263
7.6251	31.319
7.1907	31.356
7.046	31.412
6.9494	31.468
6.7081	31.524

Heat Flux	Skin Temperature
6.3221	31.581
6.0325	31.618
5.7429	31.674
5.5016	31.712
5.2603	31.768
5.1638	31.825
5.0673	31.862
4.9708	31.919
4.8743	31.975
4.7777	32.013
4.7295	32.07
4.633	32.126
4.4882	32.164

Heat Flux	Skin Temperature
4.3917	32.221
4.2951	32.259
4.2469	32.296
4.0056	32.334
3.716	32.391
3.523	32.448
3.4265	32.505
3.3782	32.543
3.4265	32.6
3.3782	32.657
3.3299	32.696
3.3299	32.753
3.4265	32.791

Simple Linear Regression

- The two-dimensional scatter plot shows that there is a strong linear relationship between the two variables



Measurements of heat flux and skin temperature of a person.

Least square Method

- Suppose we wish to fit the following linear model to the observed data:

  $f(x) = w_1 x + w_0 : w_1, w_0$ are **regression coefficients**

- Applying **method of least square**

  SSE $= \sum_{i=1}^{N} \left[y_i - f(x_i)\right]^2 = \sum_{i=1}^{N} \left[y_i - w_1 x_i - w_0\right]^2$

  $SSE = Sum\ of\ the\ Square\ Error$

Least square Method

- Optimizing by partial derivative with respect to w_0 & w_1

$$\frac{\partial E}{\partial w_0} = -2\sum_{i=1}^{N}\left[y_i - w_1 x_i - w_0\right] = 0$$

$$\frac{\partial E}{\partial w_1} = -2\sum_{i=1}^{N}\left[y_i - w_1 x_i - w_0\right] x_i = 0$$

- Summarizing above equations by matrix equation(**normal equation**)

$$\begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

Least square Method

- Values based on above data

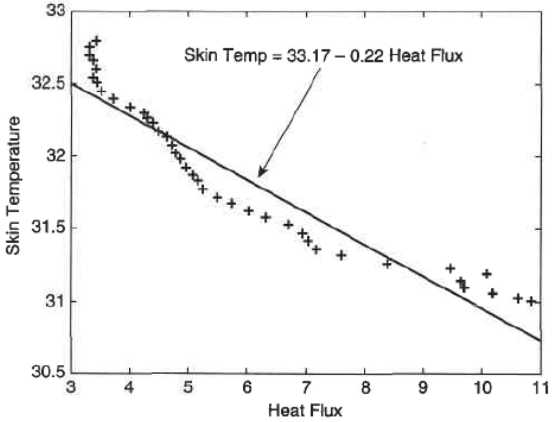
$\sum_i x_i = 229.9$, $\sum_i x_i^2 = 1569.2, \sum_i y_i = 1249.9$, $\sum_i x_i y_i = 7279.7$

$$\begin{pmatrix} \hat{w}_0 \\ \hat{w}_1 \end{pmatrix} = \begin{pmatrix} 39 & 229.9 \\ 229.9 & 1569.2 \end{pmatrix}^{-1} \begin{pmatrix} 1242.9 \\ 7279.7 \end{pmatrix}$$

$$= \begin{pmatrix} 0.1881 & -0.0276 \\ -0.0276 & 0.0047 \end{pmatrix}^{-1} \begin{pmatrix} 1242.9 \\ 7279.7 \end{pmatrix}$$

$$= \begin{pmatrix} 33.1699 \\ -0.2208 \end{pmatrix}$$

- The linear model that best fits the data in terms of minimizing the SSE is $f(x) = 33.17 - 0.22x$



A linear model that fits the data given

- The above **normal equations** can be expressed as follows:

  $w_0 = \frac{1}{N}(\sum y_i - w_1 \sum x_i)$

  i.e, $w_0 = \bar{y} - w_1 \bar{x}$

  $w_1 = \frac{N \sum x_i y_i - \sum x_i \sum y_i}{N \sum x_i^2 - (\sum x_i)^2}$

  i.e, $w_1 = \frac{\sum x_i y_i - N \bar{x} \bar{y}}{\sum x_i^2 - N(\bar{x})^2}$

Least square Method

- The **normal equations** can also be expressed as follows:

$\hat{w_0} = \bar{y} - \hat{w_1}\bar{x}$

$\hat{w_1} = \sigma_{xy}/\sigma_{xx}$

$\bar{x} = \frac{\sum_i x_i}{N}$,$\bar{y} = \frac{\sum_i y_i}{N}$

$\sigma_{xy} = \sum_i (x_i - \bar{x})(y_i - \bar{y})$

$\sigma_{xx} = \sum_i (x_i - \bar{x})^2$

$\sigma_{yy} = \sum_i (y_i - \bar{y})^2$

- Linear model that results in the minimum squared error

$f(x) = \bar{y} + \frac{\sigma_{xy}}{\sigma_{xx}}[x - \bar{x}]$

Python Code:Least Square Method

Sample Python Code to implement normal equations for **Single Variate** having X-Values and Y-Values as follows:
X=[1,2,3,4,5] Y=[3,4,6,5,6]

Comment is Shown by :

```
:Importing Python Libraries
from statistics import mean
import numpy as np
import matplotlib.pyplot as plt

:Storing Value as numpy Array
xs = np.array([1,2,3,4,5], dtype=np.float64)
ys = np.array([3,4,6,5,6], dtype=np.float64)

:Mean Calculation
Ymean=mean(ys)
Xmean=mean(xs)
```

Python Code:Least Square Method

```
:sigma_xy Calculation as per equation
sigma_xy=0
for i in range(len(xs)):
temp=(xs[i]-Xmean)*(ys[i]-Ymean)
sigma_xy=sigma_xy+temp

:sigma_xx Calculation as per equation
sigma_xx=0
for i in range(len(xs)):
   temp=(xs[i]-Xmean)*(xs[i]-Xmean)
   sigma_xx=sigma_xx+temp

:Defining Scale of X-axis and Y-axis as 0 to 10 for plotting
plt.axis([0, 10, 0, 10])
```

Python Code:Least Square Method

```
:Calculation of Y Value for the given x Value
def best_fit_slope_and_intercept(x):
   m = (sigma_xy/sigma_xx)
   Y=Ymean+m*(x-Xmean)
   return Y

:Decalaring an Array having values 0,1,2,....,9
Xs=np.arange(10)

:Calculating the Predicted value for Xs array and storing in
    array Y_Pred
Y_Pred=np.zeros((10))
for i in range(len(Xs)):
   temp=Xs[i]
   Y_Pred[i]=best_fit_slope_and_intercept(temp)
```

Python Code:Least Square Method

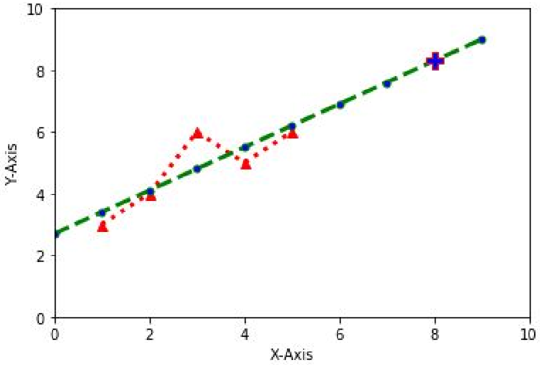
```
:Plotting the Sample Data
plt.xlabel("X-Axis") : Labelling the X-Axis
plt.ylabel("Y-Axis") : Labelling the Y-Axis
plt.plot(xs, ys, color='r', linestyle='dotted', linewidth = 3,
marker='^', markersize=7)

: Plotting the Predicted Data
plt.plot(Xs, Y_Pred, color='green', linestyle='dashed',
    linewidth = 3, marker='o', markerfacecolor='blue',
    markersize=5)
Y = best_fit_slope_and_intercept(8)
print("Predicted value of Linear Regression : ",Y )

: Marking the Predicted Point
plt.plot(8, Y, color='r', linestyle='dashed', linewidth = 3,
marker='P', markerfacecolor='blue', markersize=12)
plt.show()
```

Predicted value of Linear Regression for X-Value (8): 8.3

Python Code:Least Square Method



A Linear Model that fits the Sample Data

- The Point referred by  showing the Prediction Point

Mannual Calculation: Least Square Method

- Linear model that results in the minimum squared error

  $f(x) = \bar{y} + \frac{\sigma_{xy}}{\sigma_{xx}}[x - \bar{x}]$

- Linear model using in above program

  $f(x) = 4.8 + \frac{7.0}{10.0}[x - 3.0]$

  $f(x) = 2.7 + 0.7x$

  $f(8) = 2.7 + 0.7\text{x}8 = 8.3$

Modified Code using Co-Variance

```
from statistics import mean
import numpy as np
import matplotlib.pyplot as plt
xs = np.array([1,2,3,4,5], dtype=np.float64)
ys = np.array([3,4,6,5,6], dtype=np.float64)
Ymean=mean(ys)
Xmean=mean(xs)

temp= 4*np.cov(xs,ys)

Comment : Calling Co-Variance Library Using Parent Numpy as np
       and Storing value in matrix temp[2,2] where :
       temp[0,0]=sigma_xx temp[0,1]=sigma_xy=temp[1,0]
       temp[1,1]=sigma_yy and Cov(x, y)= sigma_xy/(n-1)
```

Python Code:Least Square Method

```
sigma_xy=temp[0,1]
sigma_xx=temp[0,0]

def best_fit_slope_and_intercept(x):
   m = (sigma_xy/sigma_xx)
   Y=Ymean+m*(x-Xmean)
   return Y

Y = best_fit_slope_and_intercept(8)
print("Predicted value of Linear Regression : ",Y )
```

Predicted value of Linear Regression : 8.3

Practice Question :

X: 2 4 6 8

Y: 3 7 5 10

Ans: $y=1.5 + 0.95x$

- The normal equations can be written in a more compact form using following matrix notations.

  Let $\mathbf{X} = \begin{pmatrix} \mathbf{1} & \mathbf{x} \end{pmatrix}$ where $\mathbf{1} = (1, 1, 1, ...)^T$ and $\mathbf{x} = (x_1, x_2, ..., x_N)^T$

  $$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} \mathbf{1}^T\mathbf{1} & \mathbf{1}^T\mathbf{x} \\ \mathbf{x}^T\mathbf{1} & \mathbf{x}^T\mathbf{x} \end{pmatrix} = \begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}$$

- if $\mathbf{y} = (y_1, y_2, ..., y_N)^T$, we can show that

  $$\begin{pmatrix} \mathbf{1} & \mathbf{x} \end{pmatrix}^T \mathbf{y} = \begin{pmatrix} \mathbf{1}^T\mathbf{y} \\ \mathbf{x}^T\mathbf{y} \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

Multivariate Linear Regression

$$\mathbf{X}^T\mathbf{X}\Omega = \mathbf{X}^T\mathbf{y} \text{ where } \Omega = (w_0, w_1)^T$$

- Parameter Ω can be solved as follows:

$$\Omega = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- If the attribute set consists of d explanatory attributes $(x_i, x_2, ..., x_d)$ **X** becomes an N x d **design matrix**:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & ... & x_{1d} \\ 1 & x_{21} & x_{22} & ... & x_{2d} \\ ... & ... & ... & ... & ... \\ 1 & x_{N1} & x_{N2} & ... & x_{Nd} \end{pmatrix}$$

while $\Omega = (w_0, w_1, ..., w_{d-1})^T$ is a d-dimensional vector.

Python Code:Multivariate Linear Regression

- The dataset for this code is taken from:
  https://drive.google.com/open?id=1mVmGNx6cbfvRHC_DvF12ZL3wGLSHD9f_

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
dataset = pd.read_csv('Filepath')
dataset.head()
X = dataset[['Petrol_tax', 'Average_income', 'Paved_Highways',
    'Population_Driver_licence(%)']]
y = dataset['Petrol_Consumption']
```

Python Code:Multivariate Linear Regression

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
print('Mean Absolute Error:',
    metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
    y_pred))
print('Root Mean Squared Error:',
    np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Output:
Mean Absolute Error: 42.26510251178464
Mean Squared Error: 2675.8793569754102
Root Mean Squared Error: 51.72890253016596