Simple Linear Regression

Rafiq Islam

2024-08-29

Table of contents

Simple Linear Regression

A simple linear regression in multiple predictors/input variables/features/independent variables/ explanatory variables/regressors/ covariates (many names) often takes the form

$$y = f(\mathbf{x}) + \epsilon = \beta \mathbf{x} + \epsilon$$

where $\beta \in \mathbb{R}^d$ are regression parameters or constant values that we aim to estimate and $\epsilon \sim \mathcal{N}(0,1)$ is a normally distributed error term independent of x or also called the white noise.

In this case, the model:

$$y = f(x) + \epsilon = \beta_0 + \beta_1 x + \epsilon$$

Therefore, in our model we need to estimate the parameters β_0, β_1. The true relationship between the explanatory variables and the dependent variable is $y = f(x)$. But our model is $y = f(x) + \epsilon$. Here, this $f(x)$ is the working model with the data. In other words, $\hat{y} = f(x) = \hat{\beta}_0 + \hat{\beta}_1 x$. Therefore, there should be some error in the model prediction which we are calling $\epsilon = \|y - \hat{y}\|$ where y is the true value and $\hat{y}$ is the predicted value. This error term is normally distributed with mean 0 and variance 1. To get the best estimate of the parameters

β_0, β_1 we can minimize the error term as much as possible. So, we define the residual sum of squares (RSS) as:

$$RSS = \epsilon_1^2 + \epsilon_2^2 + \cdots + \epsilon_{10}^2 \tag{1}$$

$$= \sum_{i=1}^{10}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{2}$$

$$\hat{\updownarrow}(\bar{\beta}) = \sum_{i=1}^{10}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{3}$$

$$\tag{4}$$

Using multivariate calculus we see

$$\frac{\partial l}{\partial \beta_0} = \sum_{i=1}^{10} 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-1) \tag{5}$$

$$\frac{\partial l}{\partial \beta_1} = \sum_{i=1}^{10} 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-x_i) \tag{6}$$

Setting the partial derivatives to zero we solve for $\hat{\beta}_0, \hat{\beta}_1$ as follows

$$\frac{\partial l}{\partial \beta_0} = 0$$

$$\implies \sum_{i=1}^{10} y_i - 10\hat{\beta}_0 - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i \right) = 0$$

$$\implies \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

and,

$$\frac{\partial l}{\partial \beta_1} = 0$$

$$\implies \sum_{i=1}^{10} 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-x_i) = 0$$

$$\implies \sum_{i=1}^{10} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(x_i) = 0$$

$$\implies \sum_{i=1}^{10} x_i y_i - \hat{\beta}_0 \left(\sum_{i=1}^{10} x_i \right) - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i^2 \right) = 0$$

$$\implies \sum_{i=1}^{10} x_i y_i - \left(\bar{y} - \hat{\beta}_1 \bar{x} \right) \left(\sum_{i=1}^{10} x_i \right) - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i^2 \right) = 0$$

$$\implies \sum_{i=1}^{10} x_i y_i - \bar{y} \left(\sum_{i=1}^{10} x_i \right) + \hat{\beta}_1 \bar{x} \left(\sum_{i=1}^{10} x_i \right) - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i^2 \right) = 0$$

$$\implies \sum_{i=1}^{10} x_i y_i - \bar{y} \left(\sum_{i=1}^{10} x_i \right) - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i^2 - \bar{x} \sum_{i=1}^{10} x_i \right) = 0$$

$$\implies \sum_{i=1}^{10} x_i y_i - \bar{y} \left(\sum_{i=1}^{10} x_i \right) - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i^2 - 10\bar{x}^2 \right) = 0$$

$$\implies \sum_{i=1}^{10} x_i y_i - \bar{y} \left(\sum_{i=1}^{10} x_i \right) - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i^2 - 2 \times 10 \times \bar{x}^2 + 10\bar{x}^2 \right) = 0$$

$$\implies \hat{\beta}_1 = \frac{\sum_{i=1}^{10} x_i y_i - 10\bar{x}\bar{y}}{\sum_{i=1}^{10} x_i^2 - 2 \times 10 \times \bar{x}^2 + 10\bar{x}^2}$$

$$\implies \hat{\beta}_1 = \frac{\sum_{i=1}^{10} x_i y_i - 10\bar{x}\bar{y} - 10\bar{x}\bar{y} + 10\bar{x}\bar{y}}{\sum_{i=1}^{10} x_i^2 - 2\bar{x} \times 10 \times \frac{1}{10} \sum_{i=1}^{10} x_i + 10\bar{x}^2}$$

$$\implies \hat{\beta}_1 = \frac{\sum_{i=1}^{10} x_i y_i - \bar{y} \left(\sum_{i=1}^{10} x_i \right) - \bar{x} \left(\sum_{i=1}^{10} y_i \right) + 10\bar{x}\bar{y}}{\sum_{i=1}^{10} (x_i - \bar{x})^2}$$

$$\implies \hat{\beta}_1 = \frac{\sum_{i=1}^{10} \left(x_i y_i - x_i \bar{y} - \bar{x} y_i + \bar{x}\bar{y} \right)}{\sum_{i=1}^{10} (x_i - \bar{x})^2}$$

$$\implies \hat{\beta}_1 = \frac{\sum_{i=1}^{10} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{10} (x_i - \bar{x})^2}$$

Therefore, we have the following

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{10}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{10}(x_i - \bar{x})^2}$$

Simple Linear Regression `slr` is applicable for a single feature data set with contineous response variable.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

Assumptions of Linear Regressions

- **Linearity:** The relationship between the feature set and the target variable has to be linear.

- **Homoscedasticity:** The variance of the residuals has to be constant.

- **Independence:** All the observations are independent of each other.

- **Normality:** The distribution of the dependent variable y has to be normal.

Synthetic Data

To implement the algorithm, we need some synthetic data. To generate the synthetic data we use the linear equation $y(x) = 2x + \frac{1}{2} + \xi$ where $\xi \sim \mathbf{N}(0,1)$

```python
X=np.random.random(100)
y=2*X+0.5+np.random.randn(100)
```

Note that we used two random number generators, `np.random.random(n)` and `np.random.randn(n)`. The first one generates n random numbers of values from the range (0,1) and the second one generates values from the standard normal distribution with mean 0 and variance or standard deviation 1.

```
plt.figure(figsize=(9,6))
plt.scatter(X,y)
plt.xlabel('$X$')
plt.ylabel('y')
plt.gca().set_facecolor('#f4f4f4')
plt.gcf().patch.set_facecolor('#f4f4f4')
plt.show()
```
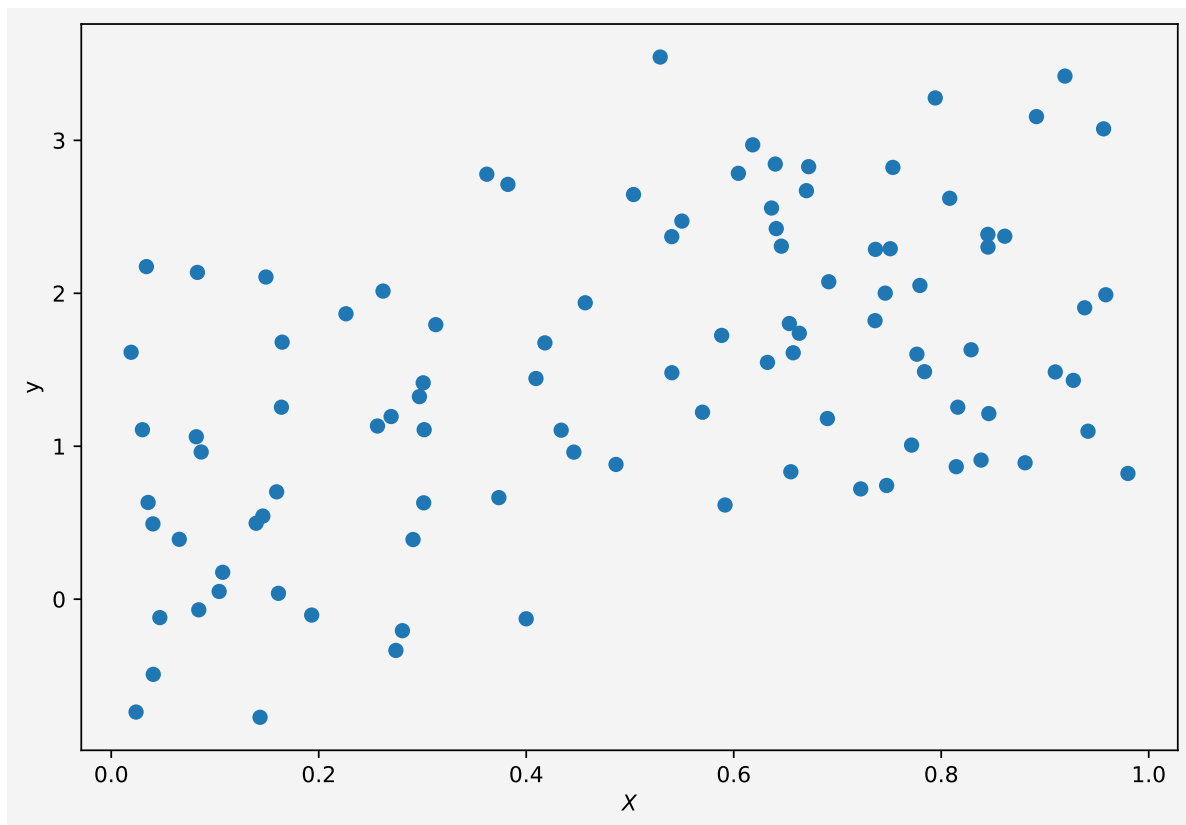

Model

We want to fit a simple linear regression to the above data.

```
slr=LinearRegression()
```

Now to fit our data X and y we need to reshape the input variable. Because if we look at X,

```
X
```

```
array([0.66321525, 0.61829285, 0.80817306, 0.29090277, 0.86124075,
       0.6364356 , 0.3620193 , 0.95649976, 0.56988564, 0.65361886,
       0.26198915, 0.65732797, 0.08669425, 0.82879498, 0.04683123,
       0.69159873, 0.81449695, 0.03553095, 0.31281265, 0.79423956,
       0.26976662, 0.84585824, 0.15942932, 0.73621075, 0.77143189,
       0.14342203, 0.5291292 , 0.40937568, 0.69029295, 0.14911791,
       0.16403595, 0.77944998, 0.9382544 , 0.67006704, 0.89178595,
       0.01916708, 0.67222296, 0.64005785, 0.38234221, 0.50342941,
       0.30069969, 0.92734393, 0.04050052, 0.08307131, 0.77654385,
       0.39996536, 0.81596588, 0.55003156, 0.30162302, 0.84497578,
       0.73661558, 0.60449041, 0.54039522, 0.29704366, 0.16473544,
       0.75086575, 0.95864832, 0.30117622, 0.97995846, 0.16117828,
       0.03005568, 0.2262336 , 0.64097951, 0.25662288, 0.08198663,
       0.02390809, 0.03389042, 0.84508069, 0.06558561, 0.37364278,
       0.19319807, 0.48654347, 0.27437197, 0.45686824, 0.91927709,
       0.08438103, 0.64587625, 0.9099459 , 0.43367222, 0.75341582,
       0.58836655, 0.13967714, 0.04017826, 0.83839688, 0.7473877 ,
       0.5402834 , 0.94148148, 0.59162281, 0.10742983, 0.7224163 ,
       0.1040249 , 0.14611912, 0.78392146, 0.44590117, 0.74603031,
       0.6325278 , 0.88083481, 0.65504465, 0.28064627, 0.41798703])
```

It is a one-dimensional array/vector but the `slr` object accepts input variable as matrix or two-dimensional format.

```
X=X.reshape(-1,1)
X[:10]
```

```
array([[0.66321525],
       [0.61829285],
       [0.80817306],
       [0.29090277],
       [0.86124075],
       [0.6364356 ],
       [0.3620193 ],
       [0.95649976],
       [0.56988564],
       [0.65361886]])
```

Now we fit the data to our model

```
slr.fit(X,y)
slr.predict([[2],[3]])
```

```
array([3.98838511, 5.66734629])
```

We have our $X = 2, 3$ and the corresponding y values are from the above cell output, which are pretty close to the model $y = 2x + \frac{1}{2}$.

```
intercept = round(slr.intercept_,4)
slope = slr.coef_
```

Now our model parameters are: intercept $\beta_0 = 0.6305$ and slope $\beta_1 = \text{array}([1.67896118])$.

```
plt.figure(figsize=(9,6))
plt.scatter(X,y, alpha=0.7,label="Sample Data")
plt.plot(np.linspace(0,1,100),
    slr.predict(np.linspace(0,1,100).reshape(-1,1)),
    'k',
    label='Model $\hat{f}$'
)
plt.plot(np.linspace(0,1,100),
    2*np.linspace(0,1,100)+0.5,
    'r--',
    label='$f$'
)
plt.xlabel('$X$')
plt.ylabel('y')
plt.legend(fontsize=10)
plt.gca().set_facecolor('#f4f4f4')
plt.gcf().patch.set_facecolor('#f4f4f4')
plt.show()
```
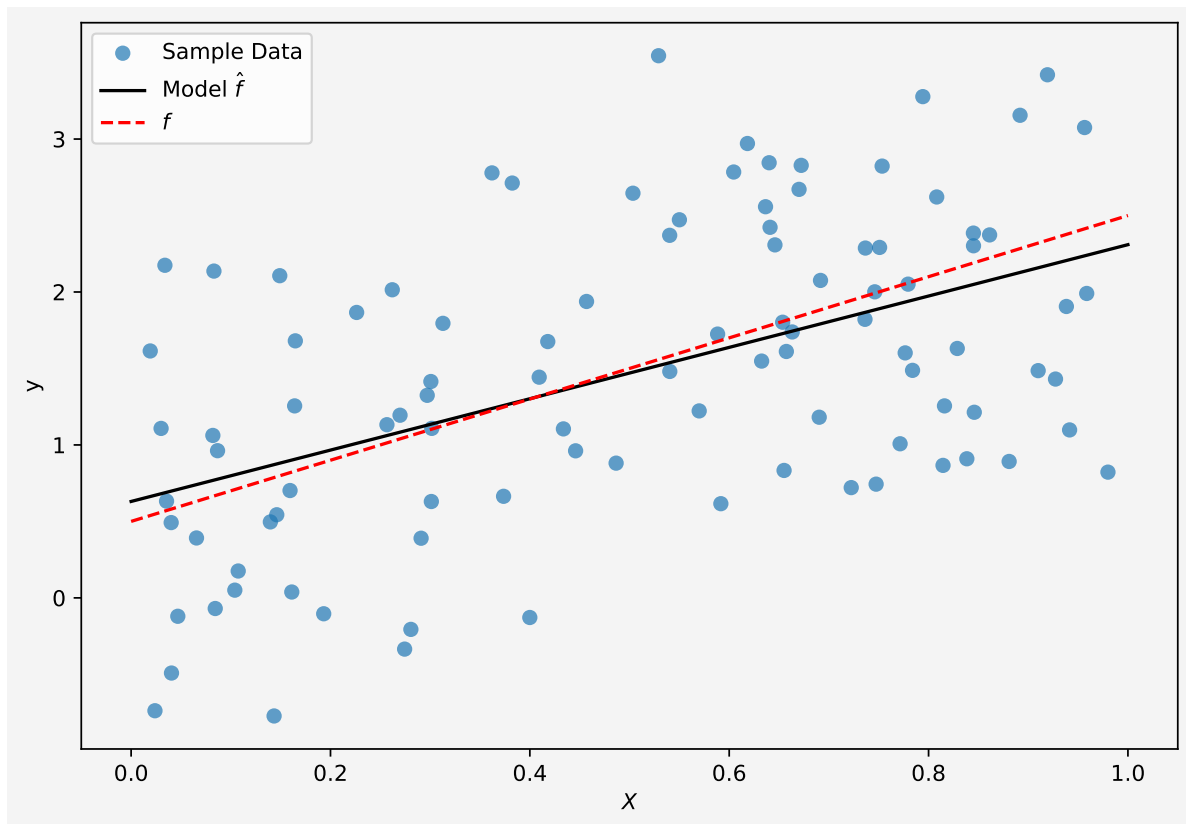
So the model fits the data almost perfectly.

Up next multiple linear regression.

Share on

f

in

y

You may also like