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,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

```python
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()
```
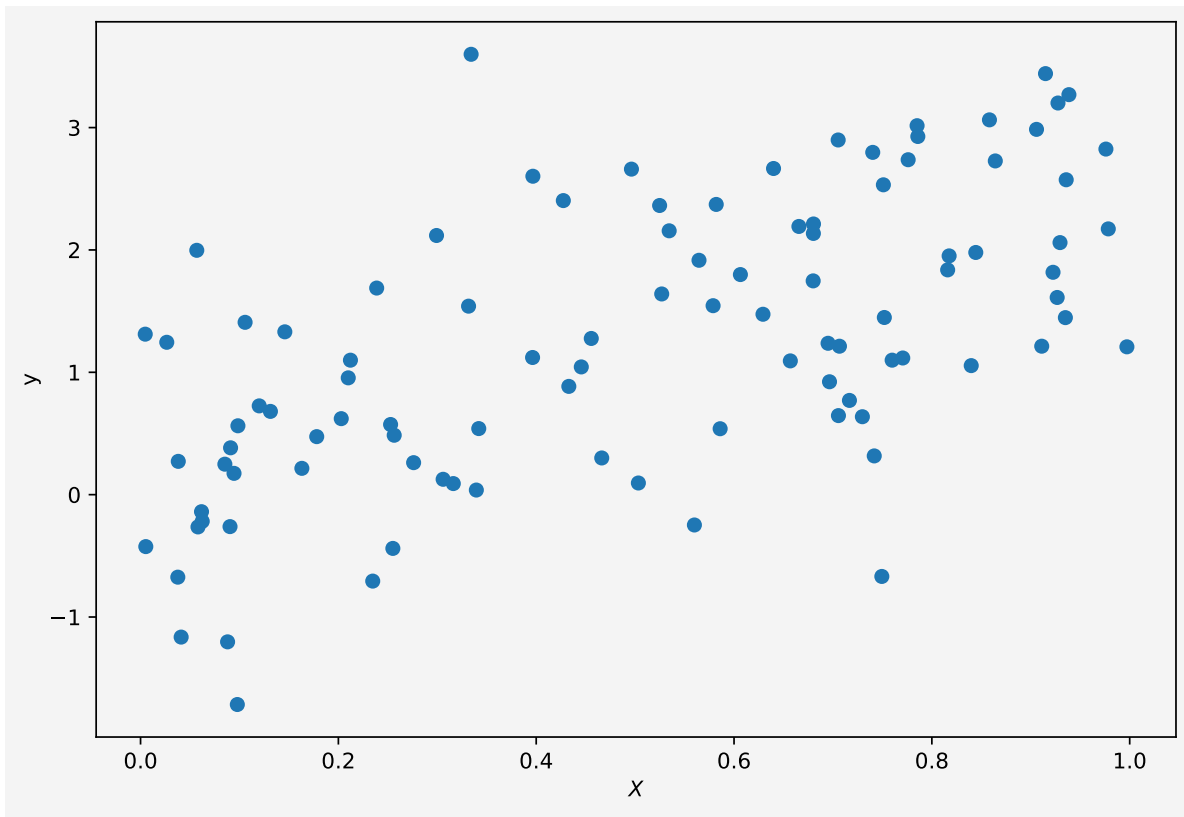

Model

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

```python
slr=LinearRegression()
```

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

5

```
X
```

```
array([0.57882425, 0.86413286, 0.03814999, 0.75098532, 0.03767413,
       0.4329759 , 0.92955845, 0.13129434, 0.17810556, 0.23872283,
       0.74022723, 0.7598504 , 0.53442797, 0.52688698, 0.58199535,
       0.74941953, 0.29922575, 0.93581208, 0.33420162, 0.93856162,
       0.77057526, 0.0568484 , 0.21219804, 0.20290498, 0.8582222 ,
       0.0879898 , 0.97596483, 0.5032937 , 0.93498195, 0.77603641,
       0.00474156, 0.44557966, 0.68020941, 0.92249008, 0.39663761,
       0.5859572 , 0.55993033, 0.69505944, 0.70527858, 0.1056939 ,
       0.08524327, 0.68031919, 0.75194546, 0.60643501, 0.2564363 ,
       0.92668965, 0.09045106, 0.62926824, 0.3963272 , 0.05806852,
       0.97835655, 0.49628669, 0.81744957, 0.09784168, 0.42741762,
       0.65691161, 0.09449506, 0.91106427, 0.02650253, 0.7858123 ,
       0.90579479, 0.31617508, 0.1630922 , 0.331613  , 0.14586334,
       0.34192144, 0.66558413, 0.20998075, 0.72975145, 0.70657378,
       0.04104996, 0.25508156, 0.99717387, 0.68000535, 0.33947692,
       0.45567186, 0.71667191, 0.27606334, 0.91488708, 0.09101179,
       0.78506032, 0.84439704, 0.83979559, 0.00530647, 0.81596452,
       0.46625735, 0.0616341 , 0.30590057, 0.11997369, 0.56457958,
       0.25278686, 0.70555224, 0.5247668 , 0.0623579 , 0.09843888,
       0.2347269 , 0.92750322, 0.69652271, 0.74174477, 0.63992763])
```

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([[0.57882425],
       [0.86413286],
       [0.03814999],
       [0.75098532],
       [0.03767413],
       [0.4329759 ],
       [0.92955845],
       [0.13129434],
       [0.17810556],
       [0.23872283]])
```

Now we fit the data to our model

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

```
array([4.71620557, 7.01992659])
```

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 $\beta_0 = 0.1088$ and slope $\beta_1 = \text{array}([2.30372102])$.

```
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()
```
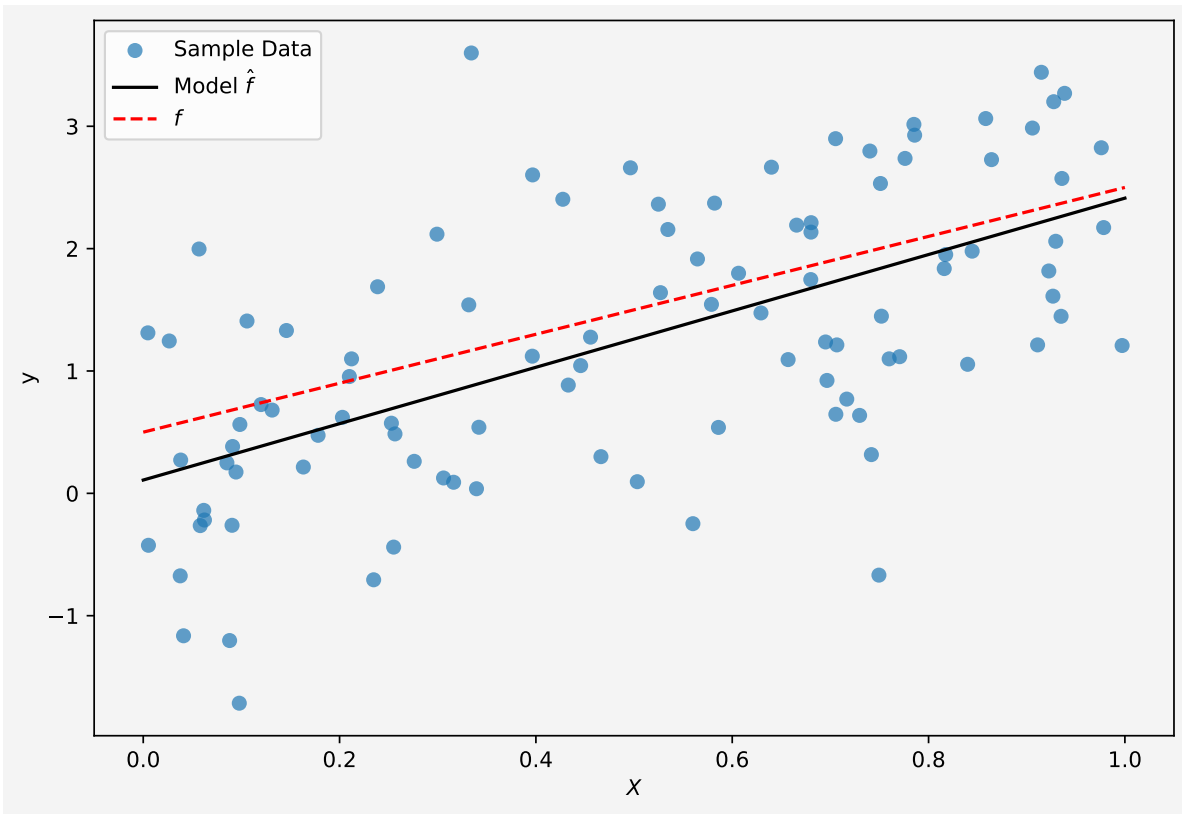
So the model fits the data almost perfectly.

Up next multiple linear regression.

Share on

You may also like