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:

$$
\begin{align}
RSS &= \epsilon_1^2 + \epsilon_2^2 + \cdots + \epsilon_{10}^2 \tag{1}\\
&= \sum_{i=1}^{10} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{2}\\
\hat{\updownarrow}(\bar{\beta}) &= \sum_{i=1}^{10} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{3}\\
& \tag{4}
\end{align}
$$

Using multivariate calculus we see

$$
\begin{align}
\frac{\partial l}{\partial \beta_0} &= \sum_{i=1}^{10} 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-1) \tag{5}\\
\frac{\partial l}{\partial \beta_1} &= \sum_{i=1}^{10} 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-x_i) \tag{6}
\end{align}
$$

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

$$
\begin{aligned}
\frac{\partial l}{\partial \beta_0} &= 0\\
\implies \sum_{i=1}^{10} y_i - 10\hat{\beta}_0 - \hat{\beta}_1 \left(\sum_{i=1}^{10} x_i \right) &= 0\\
\implies \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x}
\end{aligned}
$$

and,

$$
\begin{aligned}
\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}
\end{aligned}
$$

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.

```
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.

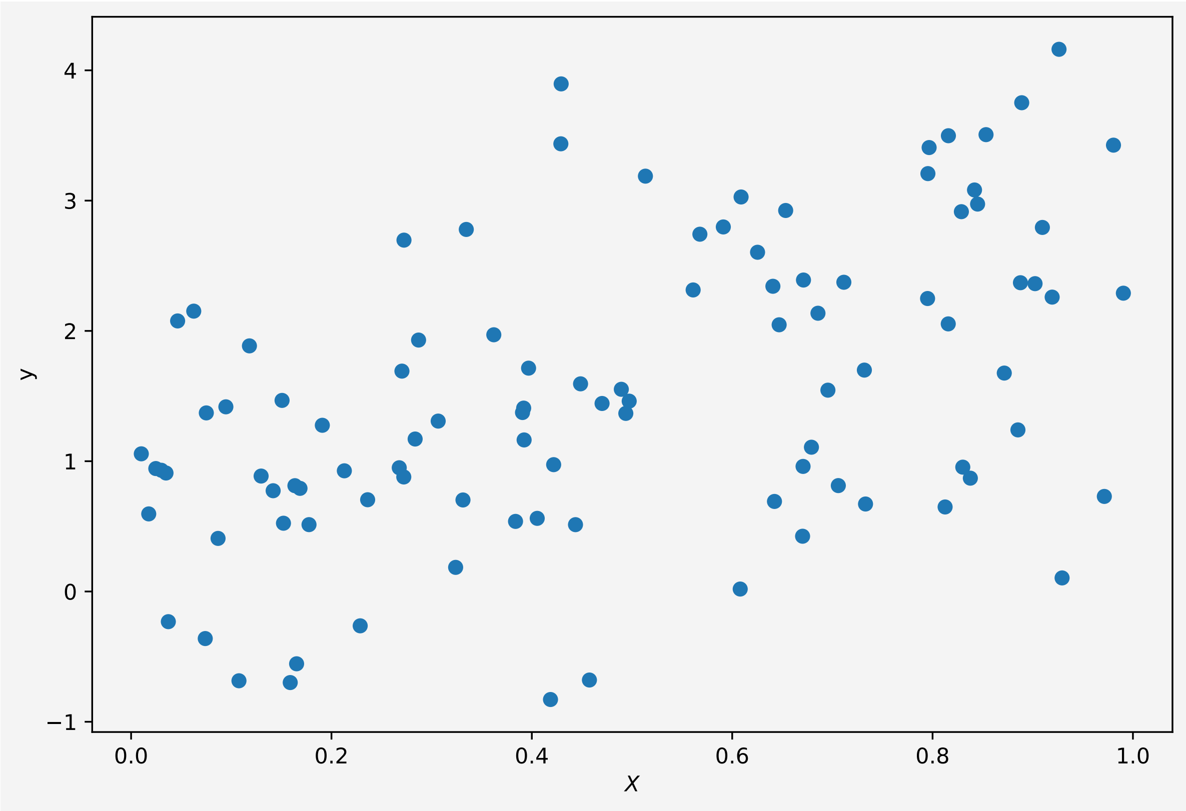
Synthetic Data

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 $\xi \sim \mathbf{N}(0,1)$

```
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,

```python
X
```

```
array([0.14178866, 0.87157493, 0.2676522 , 0.88899976, 0.84494153,
       0.56101455, 0.91937302, 0.81581116, 0.44357678, 0.5133753 ,
       0.79501693, 0.16351339, 0.15068325, 0.09459451, 0.2286768 ,
       0.85336171, 0.60886518, 0.12977285, 0.03021966, 0.39069212,
       0.27217961, 0.27033774, 0.42178116, 0.70592393, 0.4486291 ,
       0.41862348, 0.60799036, 0.07507607, 0.90229707, 0.67025524,
       0.65337559, 0.19087622, 0.27238161, 0.49380657, 0.67114856,
       0.0867917 , 0.84187162, 0.90961241, 0.99045522, 0.59112488,
       0.73307833, 0.69553854, 0.64685731, 0.97144588, 0.64062318,
       0.0348325 , 0.10766027, 0.39674703, 0.42927151, 0.68560597,
       0.81565919, 0.5677438 , 0.48932644, 0.15883789, 0.83770646,
       0.92928705, 0.33126676, 0.32393907, 0.23611238, 0.45748355,
       0.92622327, 0.67908592, 0.17758   , 0.42896466, 0.2128539 ,
       0.01017767, 0.81252143, 0.79656403, 0.6253068 , 0.73192339,
       0.33457049, 0.04643522, 0.88520648, 0.38372247, 0.36204066,
       0.4053741 , 0.82875741, 0.28352083, 0.88771815, 0.02465383,
       0.64219377, 0.47010063, 0.03716672, 0.83016576, 0.71144875,
       0.67073159, 0.98058567, 0.16857753, 0.39184882, 0.39230019,
       0.11811322, 0.07401236, 0.15208774, 0.49718081, 0.16519025,
       0.28695108, 0.01759611, 0.30652099, 0.06253239, 0.7953509 ])
```

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

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

```
array([[0.14178866],
       [0.87157493],
       [0.2676522 ],
       [0.88899976],
       [0.84494153],
       [0.56101455],
       [0.91937302],
       [0.81581116],
       [0.44357678],
       [0.5133753 ]])
```

Now we fit the data to our model

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

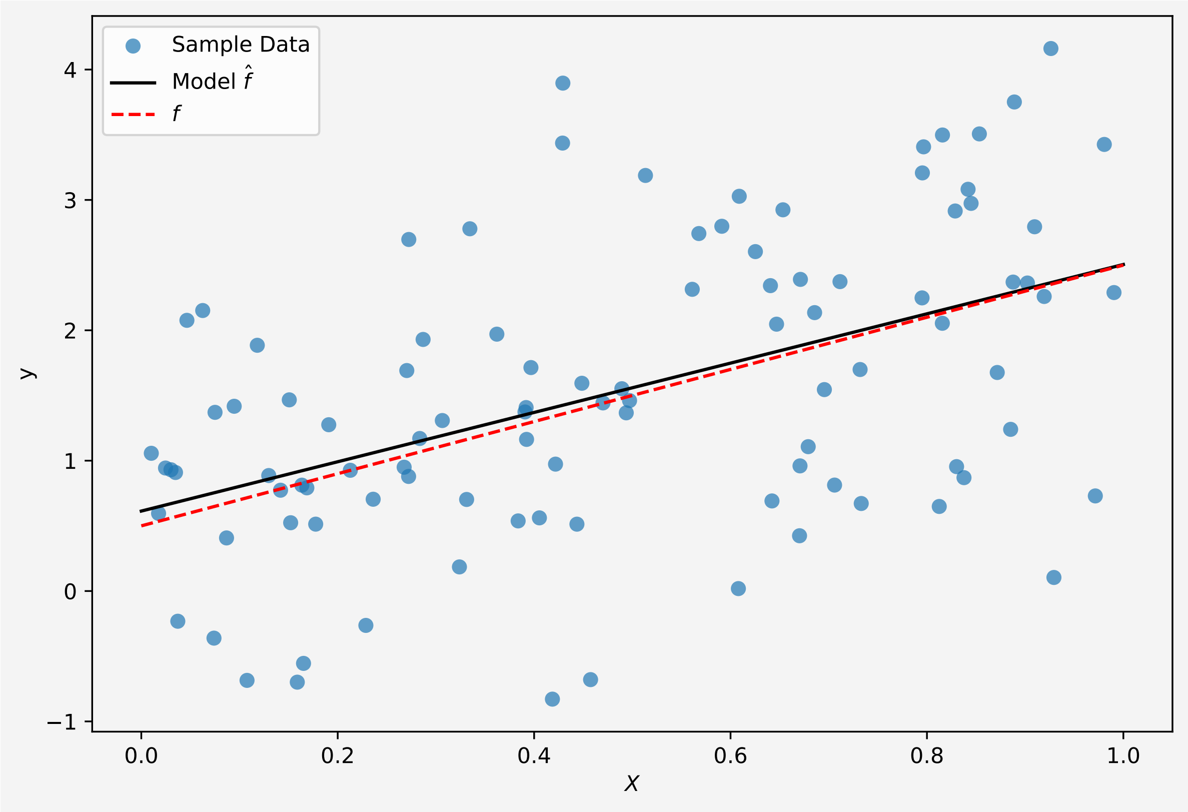
```
array([4.39614604, 6.28751252])
```

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 $\beta_0 = 0.6134$ and slope $\beta_1 = \text{array}([1.89136649])$.

```
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