

Simple Linear Regression

Rafiq Islam

2024-08-29

Table of contents

Simple Linear Regression	1
Assumptions of Linear Regressions	4
Synthetic Data	4
Model	5

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:

$$RSS = \epsilon_1^2 + \epsilon_2^2 + \cdots + \epsilon_{10}^2 \quad (1)$$

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

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

$$(4)$$

Using multivariate calculus we see

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

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

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

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

and,

$$\begin{aligned}
& \frac{\partial l}{\partial \beta_1} = 0 \\
& \Rightarrow \sum_{i=1}^{10} 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-x_i) = 0 \\
& \Rightarrow \sum_{i=1}^{10} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(x_i) = 0 \\
& \Rightarrow \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 \\
& \Rightarrow \sum_{i=1}^{10} x_i y_i - (\bar{y} - \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 \\
& \Rightarrow \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 \\
& \Rightarrow \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 \\
& \Rightarrow \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 \\
& \Rightarrow \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 \\
& \Rightarrow \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} \\
& \Rightarrow \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} \\
& \Rightarrow \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} \\
& \Rightarrow \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} \\
& \Rightarrow \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 continuous 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.

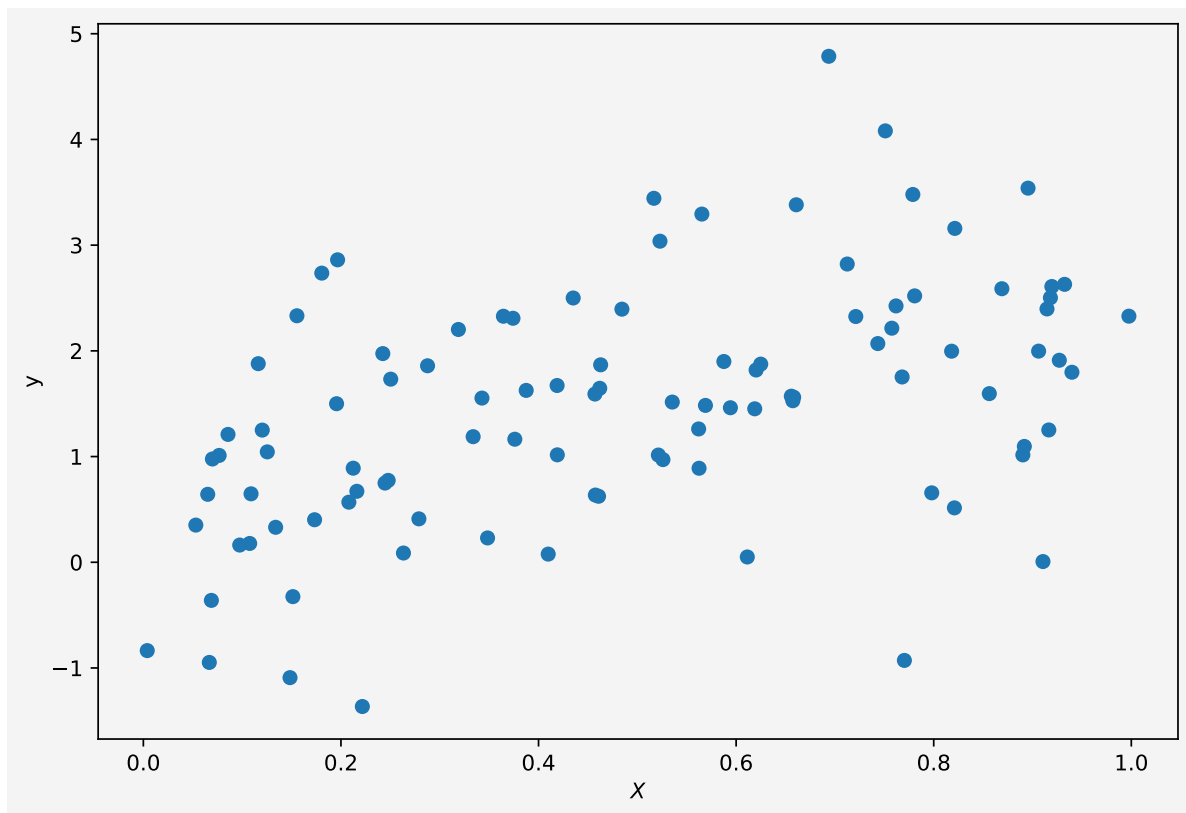
Synthetic Data

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 ,

```
X
```

```
array([0.48441344, 0.1162953 , 0.20801345, 0.56532148, 0.2124221 ,
       0.62492011, 0.8954813 , 0.37413731, 0.52291459, 0.18059733,
       0.91057446, 0.72112136, 0.86897934, 0.27890431, 0.4350852 ,
       0.33381135, 0.71246774, 0.56894274, 0.61130896, 0.89171978,
       0.15135858, 0.90618202, 0.99756048, 0.38750936, 0.00393221,
       0.81812163, 0.45752175, 0.51675691, 0.40979885, 0.46287516,
       0.24237531, 0.12035242, 0.06516011, 0.8902162 , 0.9164896 ,
       0.26327278, 0.79800167, 0.91465487, 0.24461862, 0.13389443,
       0.77886848, 0.41879746, 0.1484413 , 0.74341705, 0.34273259,
       0.85627156, 0.28765987, 0.52592945, 0.24781453, 0.36444617,
       0.19660836, 0.37599224, 0.34840214, 0.61883604, 0.65822817,
       0.76184838, 0.10762239, 0.22167038, 0.19557313, 0.93248418,
       0.77031492, 0.21610744, 0.62022264, 0.07672214, 0.2503908 ,
       0.75758246, 0.52128753, 0.15543792, 0.46195404, 0.9398995 ,
       0.09753841, 0.75102472, 0.06675663, 0.91957989, 0.17332121,
       0.0688478 , 0.66089818, 0.92712173, 0.56206387, 0.46074214,
       0.08568155, 0.58760025, 0.78069537, 0.06991455, 0.65586402,
       0.5353857 , 0.8209577 , 0.41901027, 0.65739811, 0.31888565,
       0.45706903, 0.91812104, 0.05309993, 0.6937705 , 0.59421114,
       0.12544674, 0.10899787, 0.7680258 , 0.82136467, 0.56247938])
```

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([[0.48441344],
       [0.1162953 ],
       [0.20801345],
       [0.56532148],
       [0.2124221 ],
       [0.62492011],
       [0.8954813 ],
       [0.37413731],
       [0.52291459],
       [0.18059733]])
```

Now we fit the data to our model

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

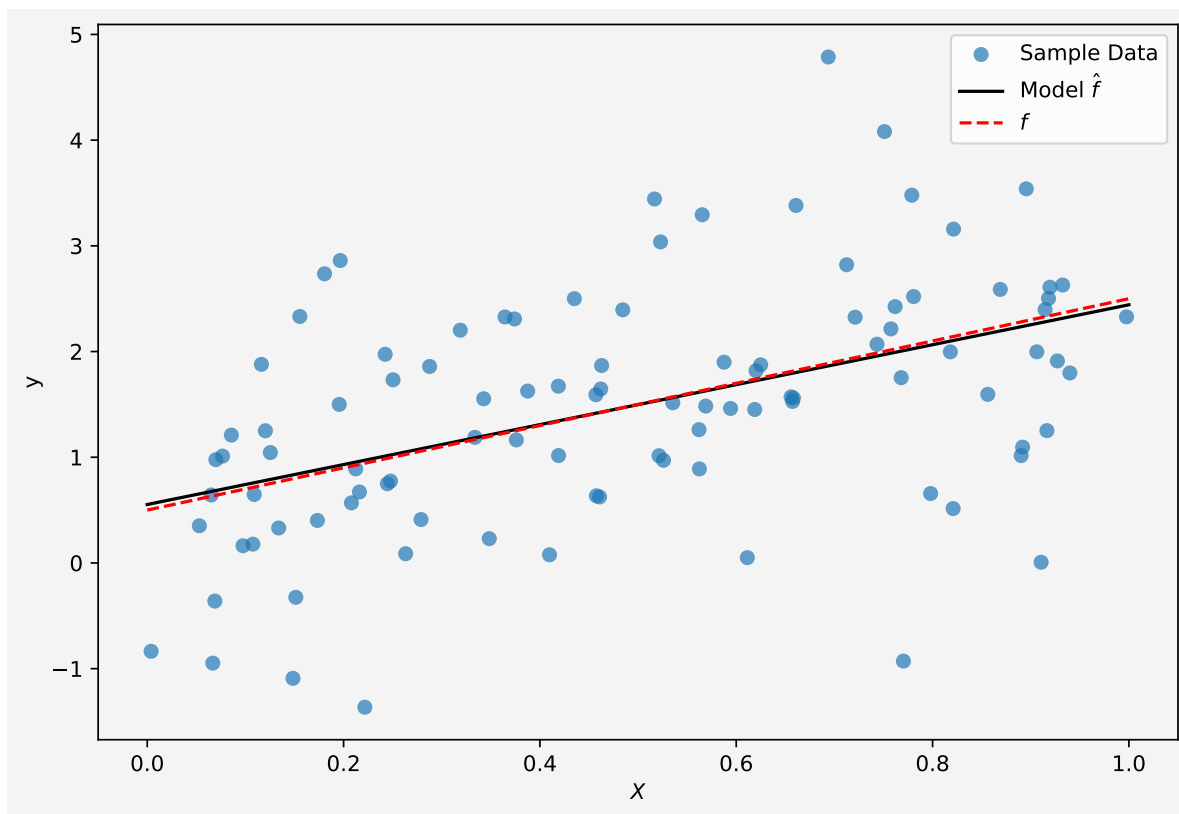
```
array([4.33155954, 6.220844  ])
```

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 $\beta_0 = 0.553$ and slope $\beta_1 = \text{array}([1.88928446])$.

```
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