

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.

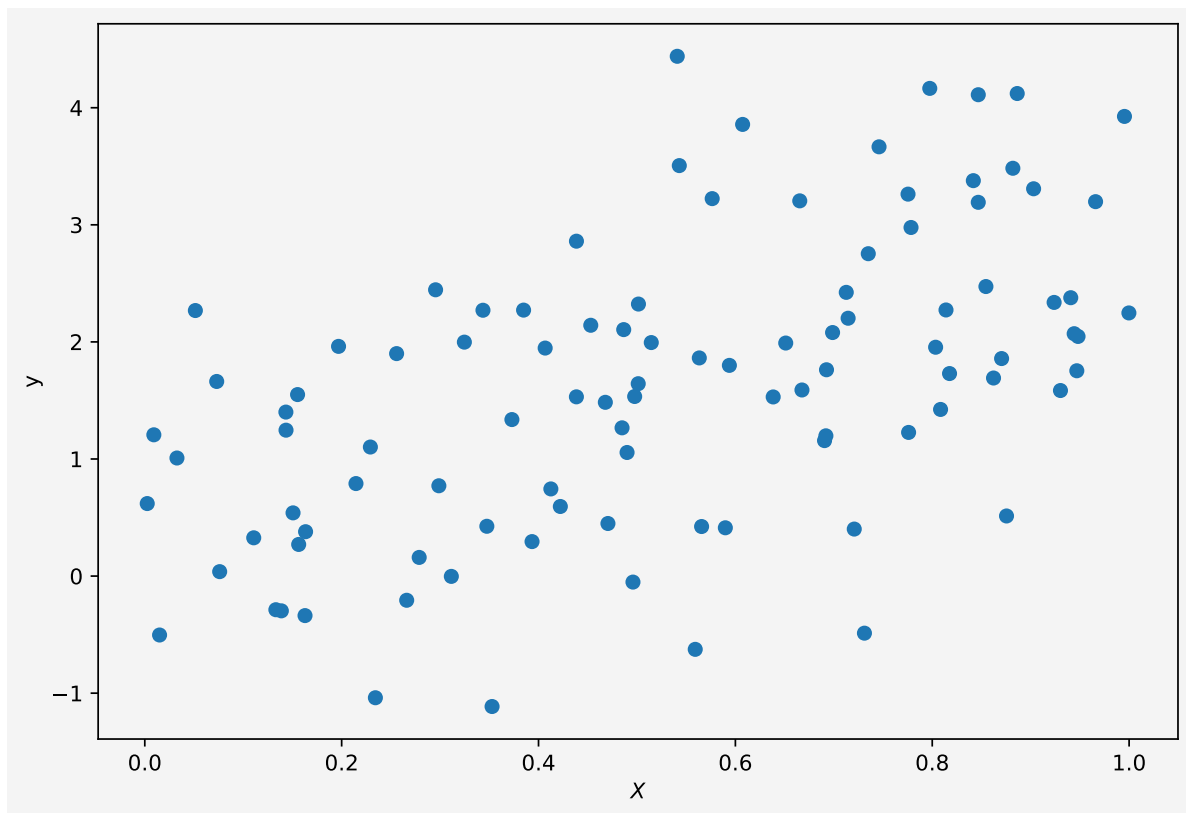
Synthetic Data

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([0.65114658, 0.15521615, 0.73111312, 0.40675412, 0.05138183,
       0.90296959, 0.29542445, 0.71264063, 0.48654823, 0.55921253,
       0.19684427, 0.63840793, 0.92380936, 0.48995117, 0.14334593,
       0.72076073, 0.84675999, 0.80844417, 0.07305464, 0.56567172,
       0.94686622, 0.8704596 , 0.74594225, 0.29876129, 0.49767667,
       0.7753934 , 0.81757081, 0.85452705, 0.99527959, 0.15633937,
       0.99978158, 0.94418667, 0.80342185, 0.69256723, 0.22918175,
       0.50129228, 0.15061015, 0.49594928, 0.93025964, 0.27876877,
       0.48486119, 0.21454611, 0.32466706, 0.4705252 , 0.96592898,
       0.54093722, 0.0326767 , 0.57643911, 0.77597288, 0.69057033,
       0.8818967 , 0.1633466 , 0.00248791, 0.54302211, 0.60734702,
       0.41253886, 0.94071706, 0.45307396, 0.66764114, 0.56333948,
       0.84173774, 0.69195225, 0.01501851, 0.35281004, 0.162785 ,
       0.88632805, 0.07607387, 0.26607023, 0.38482697, 0.31142504,
       0.1331638 , 0.73500576, 0.4678802 , 0.86225321, 0.51455078,
       0.42216564, 0.39342003, 0.14348239, 0.87546087, 0.3435646 ,
       0.5897943 , 0.77844337, 0.84674854, 0.43853546, 0.34750084,
       0.2558282 , 0.81397548, 0.00919784, 0.79748145, 0.23427392,
       0.50152042, 0.66540128, 0.43848489, 0.69874077, 0.5939023 ,
       0.11067522, 0.71453245, 0.37296946, 0.94810404, 0.13873583])
```

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([[0.65114658],
       [0.15521615],
       [0.73111312],
       [0.40675412],
       [0.05138183],
       [0.90296959],
       [0.29542445],
       [0.71264063],
       [0.48654823],
       [0.55921253]])
```

Now we fit the data to our model

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

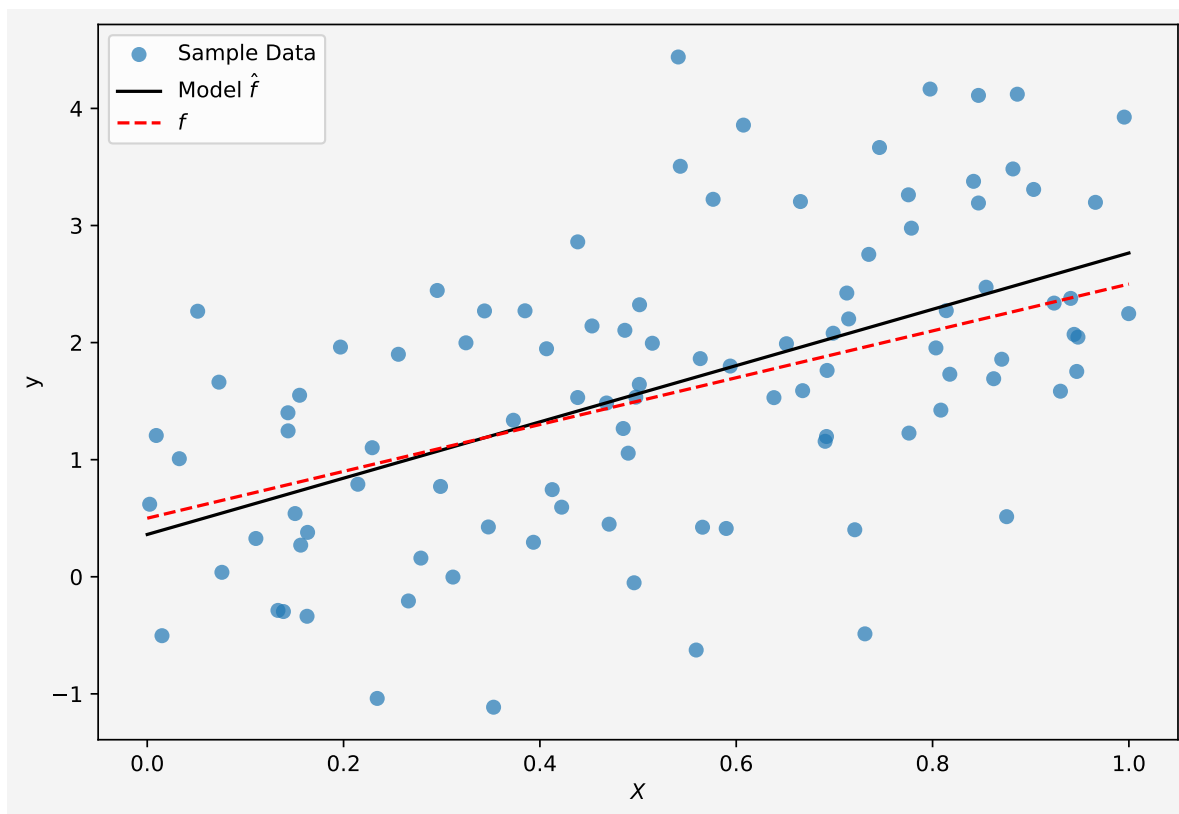
```
array([5.16884987, 7.57262182])
```

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 $\beta_0 = 0.3613$ and slope $\beta_1 = \text{array}([2.40377195])$.

```
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