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

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

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.

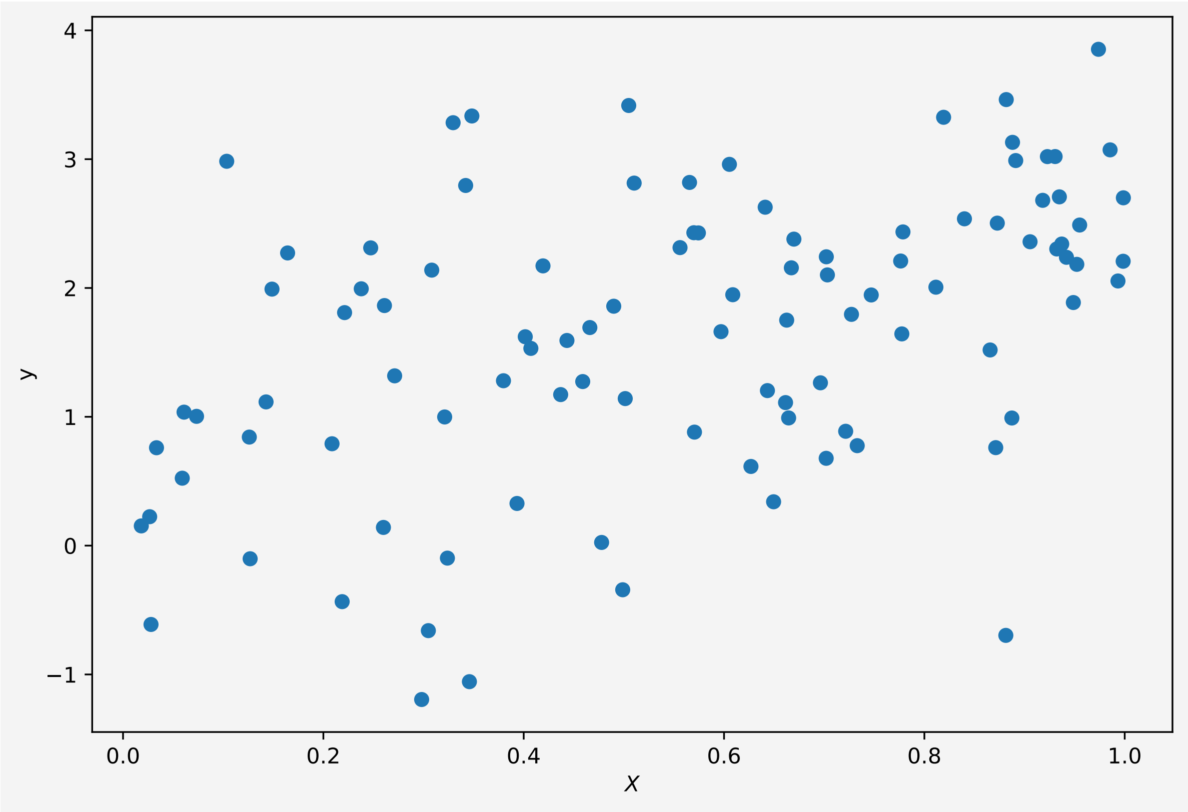
Synthetic Data

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([0.02658815, 0.94182768, 0.50485294, 0.25995786, 0.14278522,
       0.12690189, 0.3457989 , 0.018233  , 0.99844824, 0.66140301,
       0.21876757, 0.59698024, 0.74695115, 0.8817438 , 0.93051841,
       0.30480851, 0.81150564, 0.34824909, 0.77862288, 0.64947216,
       0.32383548, 0.64330604, 0.4898848 , 0.5102654 , 0.62686668,
       0.02793138, 0.66253801, 0.23782435, 0.88799489, 0.57047152,
       0.60536073, 0.7272134 , 0.64108252, 0.98541128, 0.07338382,
       0.24725484, 0.66447625, 0.26097017, 0.32953028, 0.22117923,
       0.9228127 , 0.72143441, 0.39330478, 0.57439075, 0.27115892,
       0.83999631, 0.03341056, 0.8813222 , 0.60871366, 0.93711932,
       0.32109673, 0.89120217, 0.37979812, 0.43690931, 0.06090381,
       0.20864063, 0.99326863, 0.46604017, 0.94872639, 0.95217534,
       0.55606146, 0.66974822, 0.90550731, 0.81919501, 0.95502118,
       0.87280293, 0.30820174, 0.56986638, 0.45892169, 0.77633054,
       0.50135961, 0.14871304, 0.97380017, 0.99866169, 0.47778096,
       0.12605301, 0.91812537, 0.88734335, 0.16429579, 0.29812413,
       0.87125167, 0.1035157 , 0.70210093, 0.77749875, 0.70319278,
       0.40154865, 0.56555082, 0.69622439, 0.66719715, 0.05912518,
       0.70197735, 0.40714807, 0.93476248, 0.41925564, 0.49883917,
       0.34205339, 0.86563528, 0.44316479, 0.73295782, 0.93216929])
```

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([[0.02658815],
       [0.94182768],
       [0.50485294],
       [0.25995786],
       [0.14278522],
       [0.12690189],
       [0.3457989 ],
       [0.018233  ],
       [0.99844824],
       [0.66140301]])
```

Now we fit the data to our model

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

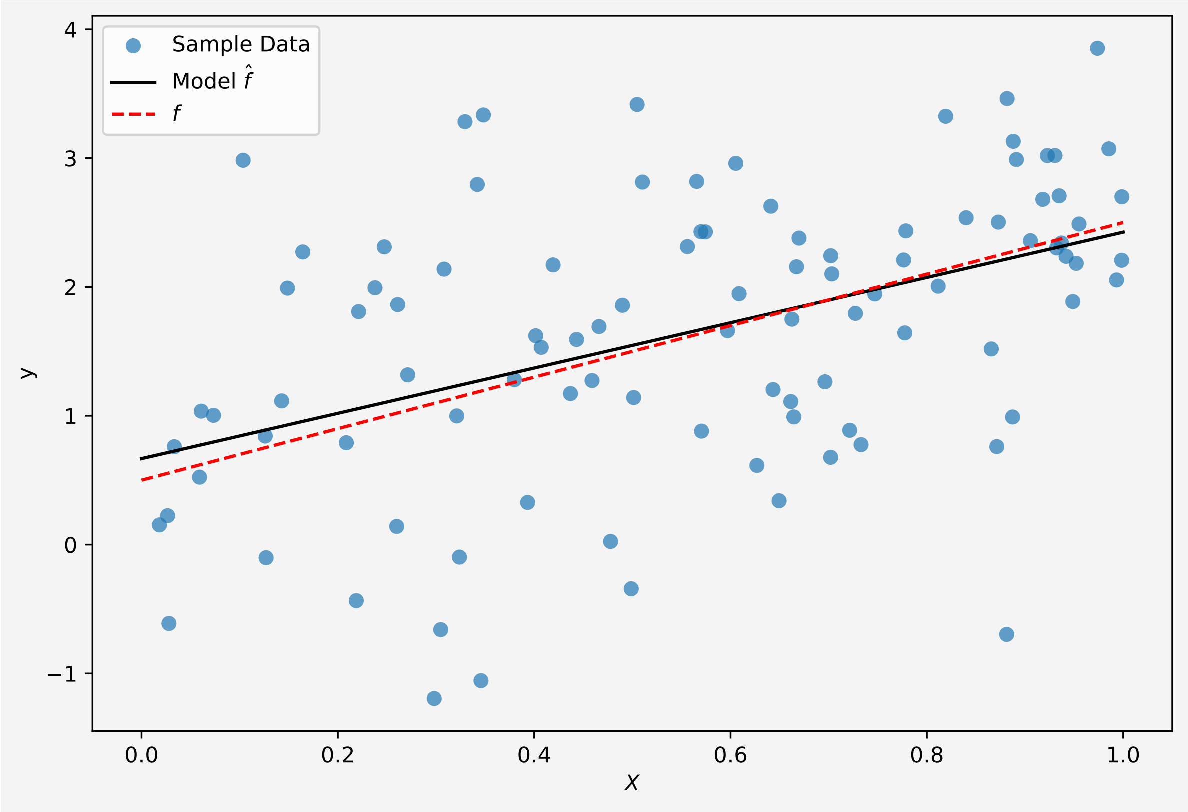
```
array([4.18364159, 5.94179303])
```

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

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

Now our model parameters are: intercept $\beta_0 = 0.6673$ and slope $\beta_1 = \text{array}([1.75815144])$.

```python
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