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:

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

Using multivariate calculus we see

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

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.

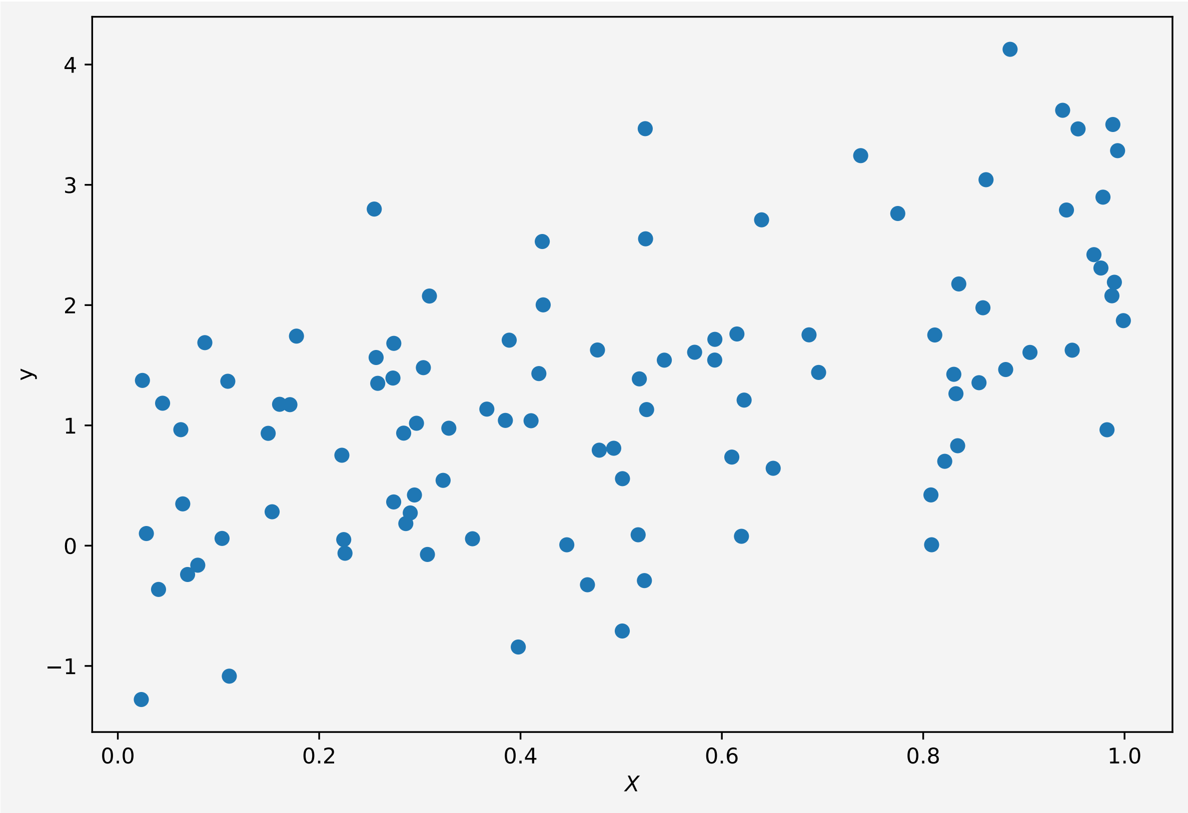
Synthetic Data

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([0.25471691, 0.22255141, 0.30755121, 0.04446896, 0.94232054,
       0.4216143 , 0.95383897, 0.17093494, 0.30952323, 0.8083695 ,
       0.14935385, 0.50108808, 0.32879717, 0.3977917 , 0.59305816,
       0.06929398, 0.51803092, 0.88631017, 0.82140494, 0.42252257,
       0.81151733, 0.62212372, 0.28393019, 0.69612346, 0.49259939,
       0.44598501, 0.3231123 , 0.63947786, 0.52388554, 0.02444274,
       0.17744884, 0.28606371, 0.11072826, 0.16066406, 0.10352972,
       0.52527694, 0.3034966 , 0.9899141 , 0.61947467, 0.52422952,
       0.22439584, 0.61496651, 0.80768173, 0.47821642, 0.47646366,
       0.41823582, 0.83041343, 0.85543676, 0.22566043, 0.41046895,
       0.07941078, 0.02331502, 0.51684832, 0.99309814, 0.27401557,
       0.52306618, 0.08644244, 0.35230912, 0.46649961, 0.38496879,
       0.15327142, 0.98259224, 0.98745789, 0.02835017, 0.50128365,
       0.1092447 , 0.86241208, 0.99883666, 0.83425016, 0.25817403,
       0.54284607, 0.83536042, 0.29047797, 0.27329421, 0.2966733 ,
       0.59291211, 0.65099991, 0.68659914, 0.572951  , 0.96959277,
       0.2565717 , 0.29464851, 0.38877179, 0.97858512, 0.9479985 ,
       0.85934443, 0.98840262, 0.88193994, 0.27414583, 0.06449421,
       0.73790924, 0.3665562 , 0.04043319, 0.97654258, 0.83252722,
       0.90606415, 0.6098945 , 0.93852278, 0.77474901, 0.06256948])
```

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([[0.25471691],
       [0.22255141],
       [0.30755121],
       [0.04446896],
       [0.94232054],
       [0.4216143 ],
       [0.95383897],
       [0.17093494],
       [0.30952323],
       [0.8083695 ]])
```

Now we fit the data to our model

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

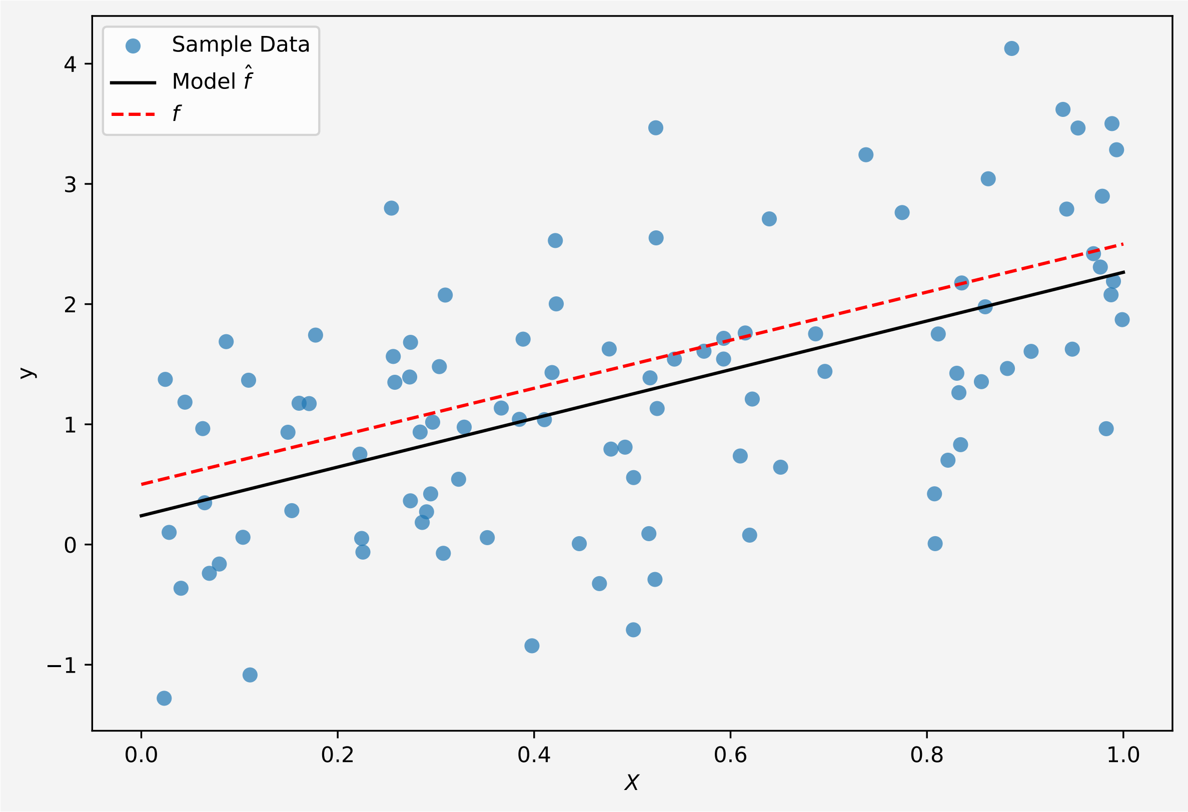
```
array([4.29001293, 6.31535518])
```

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 $\beta_0 = 0.2393$ and slope $\beta_1 = \text{array}([2.02534225])$.

```
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