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

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

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.

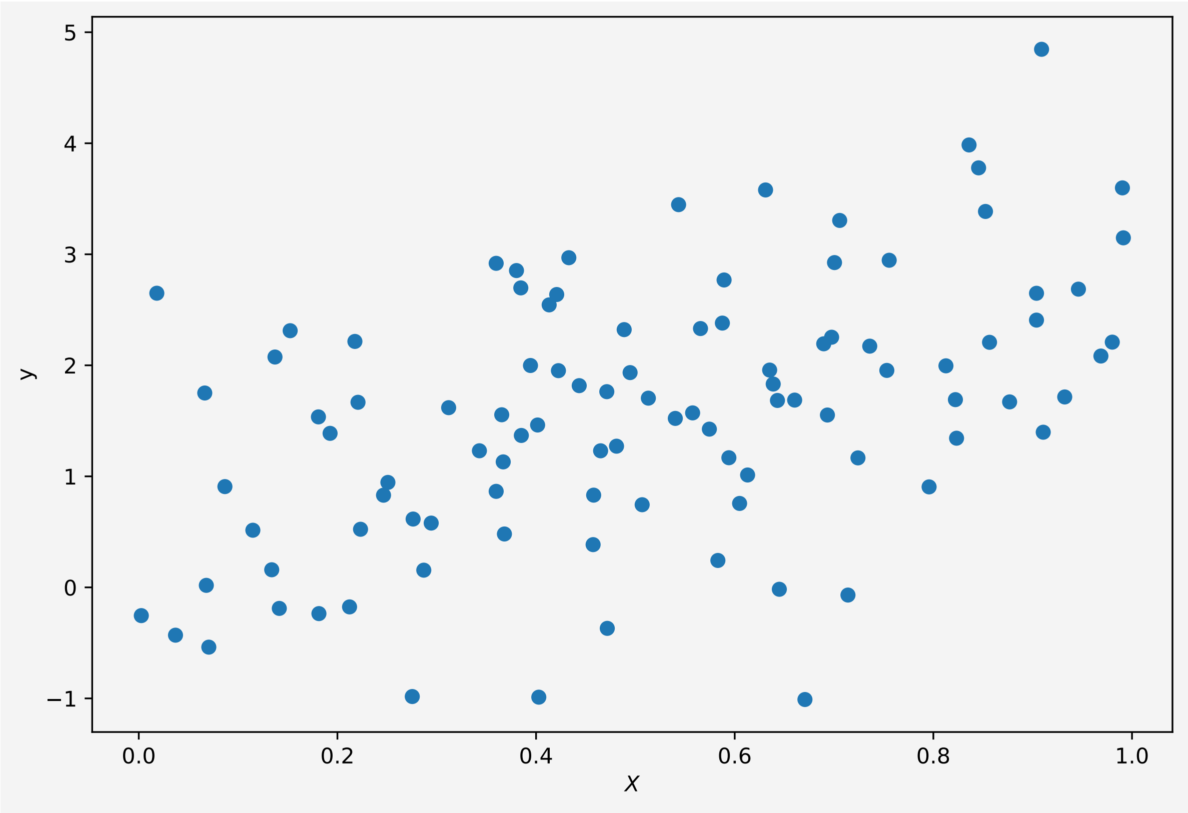
Synthetic Data

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,

```
X
```

```
array([0.18138023, 0.18088768, 0.70047574, 0.28695371, 0.41318748,
       0.46497315, 0.7531558 , 0.3598406 , 0.13714269, 0.63105914,
       0.35984139, 0.56555259, 0.21761029, 0.27621593, 0.38516649,
       0.06801113, 0.54019   , 0.21222225, 0.83593722, 0.25083007,
       0.75553928, 0.58930939, 0.13386837, 0.54351052, 0.03699006,
       0.36693266, 0.90383306, 0.08673067, 0.4887192 , 0.81266554,
       0.58757916, 0.4027564 , 0.43300035, 0.96868638, 0.01817662,
       0.60491247, 0.31199884, 0.49467695, 0.42078676, 0.15243222,
       0.51306266, 0.84554796, 0.64302582, 0.87678492, 0.57447997,
       0.64491923, 0.67071275, 0.99130285, 0.94606386, 0.22072764,
       0.70569204, 0.38028551, 0.00254717, 0.47122916, 0.11486645,
       0.73594675, 0.39437701, 0.45743698, 0.07052153, 0.79570709,
       0.34296427, 0.55762393, 0.36543981, 0.90895658, 0.29442183,
       0.40156176, 0.61297543, 0.5068252 , 0.823353  , 0.63518253,
       0.82220866, 0.90385521, 0.85655599, 0.47169842, 0.06644172,
       0.38464161, 0.19261639, 0.68962598, 0.63876692, 0.71402392,
       0.42251998, 0.69756278, 0.27537016, 0.45798368, 0.99035219,
       0.69334135, 0.58306668, 0.2232718 , 0.85248439, 0.24642396,
       0.9801408 , 0.48101234, 0.91062647, 0.66035412, 0.72409226,
       0.59413102, 0.44344357, 0.93226568, 0.36814382, 0.14148209])
```

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([[0.18138023],
       [0.18088768],
       [0.70047574],
       [0.28695371],
       [0.41318748],
       [0.46497315],
       [0.7531558 ],
       [0.3598406 ],
       [0.13714269],
       [0.63105914]])
```

Now we fit the data to our model

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

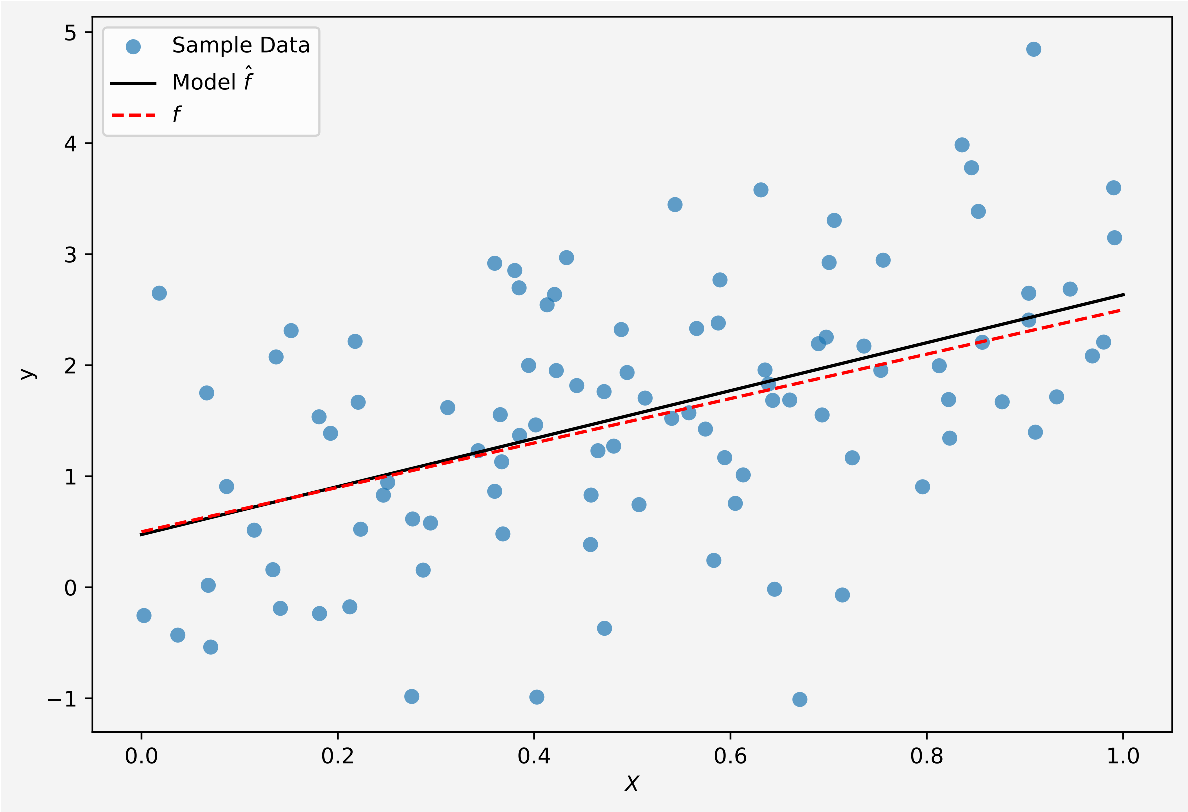
```
array([4.79470931, 6.95426483])
```

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 $\beta_0 = 0.4756$ and slope $\beta_1 = \text{array}([2.15955552])$.

```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