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.

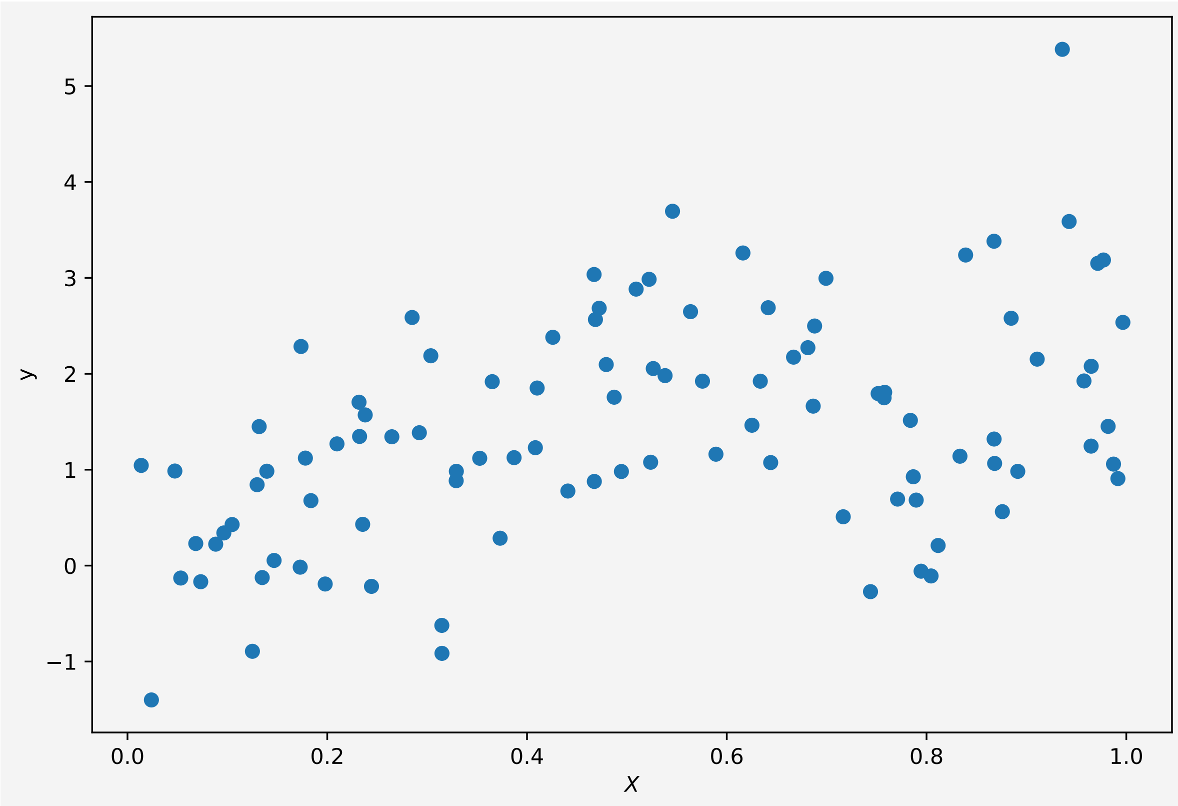
Synthetic Data

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([0.68123061, 0.10475784, 0.18365971, 0.7838254 , 0.42582459,
       0.79453229, 0.64148734, 0.86765399, 0.19793971, 0.96490147,
       0.4671514 , 0.9818086 , 0.57566827, 0.52378662, 0.13487981,
       0.91079099, 0.38705662, 0.77101224, 0.36519141, 0.9576353 ,
       0.23800208, 0.20974195, 0.14677176, 0.13183807, 0.5263912 ,
       0.0239525 , 0.6865267 , 0.97141014, 0.40835751, 0.8681827 ,
       0.78965347, 0.71660585, 0.50925203, 0.31485523, 0.64398082,
       0.30370372, 0.46738346, 0.62519905, 0.06836814, 0.83914268,
       0.29219871, 0.37306928, 0.32923617, 0.49451495, 0.48728895,
       0.56372801, 0.78674936, 0.46850746, 0.80455239, 0.07335119,
       0.44091442, 0.01376503, 0.28495932, 0.7439105 , 0.13952869,
       0.41015275, 0.6879463 , 0.23238191, 0.94277814, 0.93596065,
       0.52224561, 0.61619244, 0.66686653, 0.1737329 , 0.97712491,
       0.88473328, 0.99159798, 0.35265878, 0.12508337, 0.87592743,
       0.99659726, 0.63355176, 0.89135168, 0.98720853, 0.05331525,
       0.86756524, 0.69932836, 0.81160697, 0.1729104 , 0.53822526,
       0.31471225, 0.5456938 , 0.24429473, 0.23547401, 0.47931495,
       0.23181422, 0.32909245, 0.58916415, 0.47227595, 0.96475552,
       0.7581227 , 0.09647593, 0.08834111, 0.83346726, 0.0474393 ,
       0.1781168 , 0.75750676, 0.75154026, 0.26468221, 0.12978472])
```

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([[0.68123061],
       [0.10475784],
       [0.18365971],
       [0.7838254 ],
       [0.42582459],
       [0.79453229],
       [0.64148734],
       [0.86765399],
       [0.19793971],
       [0.96490147]])
```

Now we fit the data to our model

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

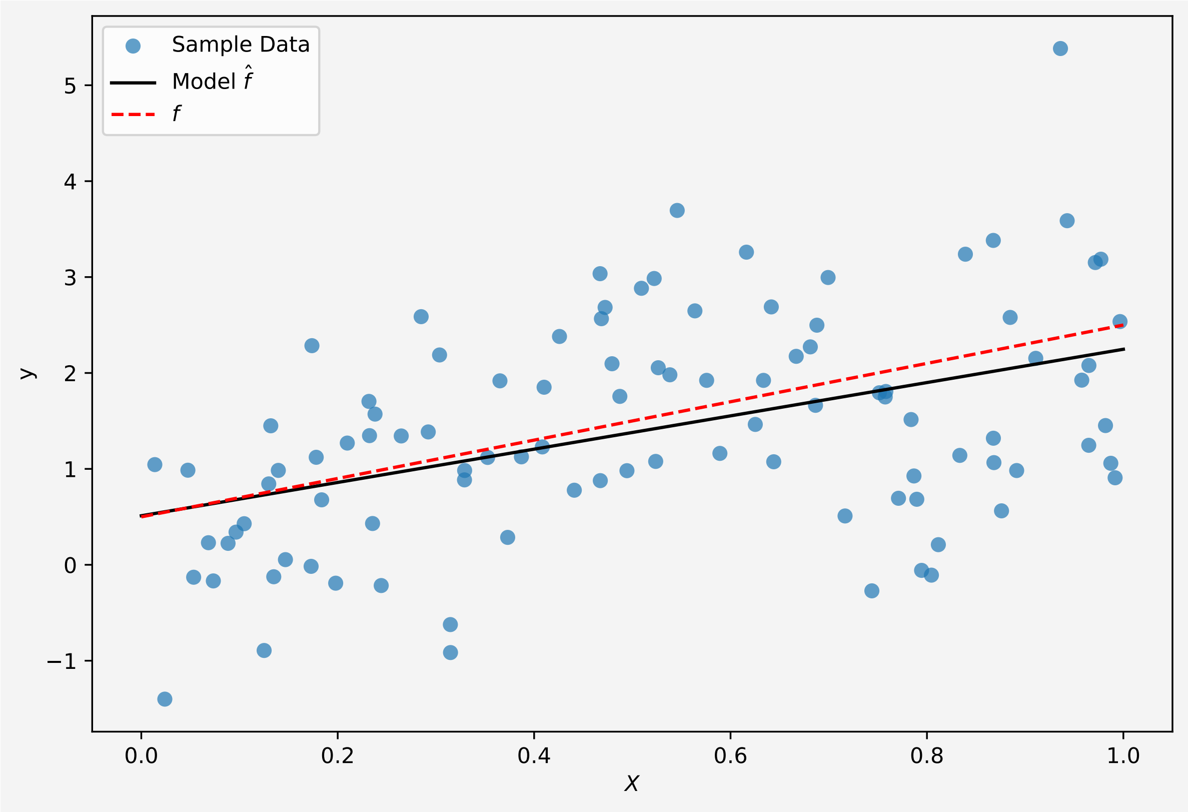
```
array([3.98293907, 5.71867186])
```

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 $\beta_0 = 0.5115$ and slope $\beta_1 = \text{array}([1.73573279])$.

```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