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:

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

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

$$
\begin{aligned}
\hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x} \\
\hat{\beta}_1 &= \frac{\sum_{i=1}^{10}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{10}(x_i - \bar{x})^2}
\end{aligned}
$$

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.

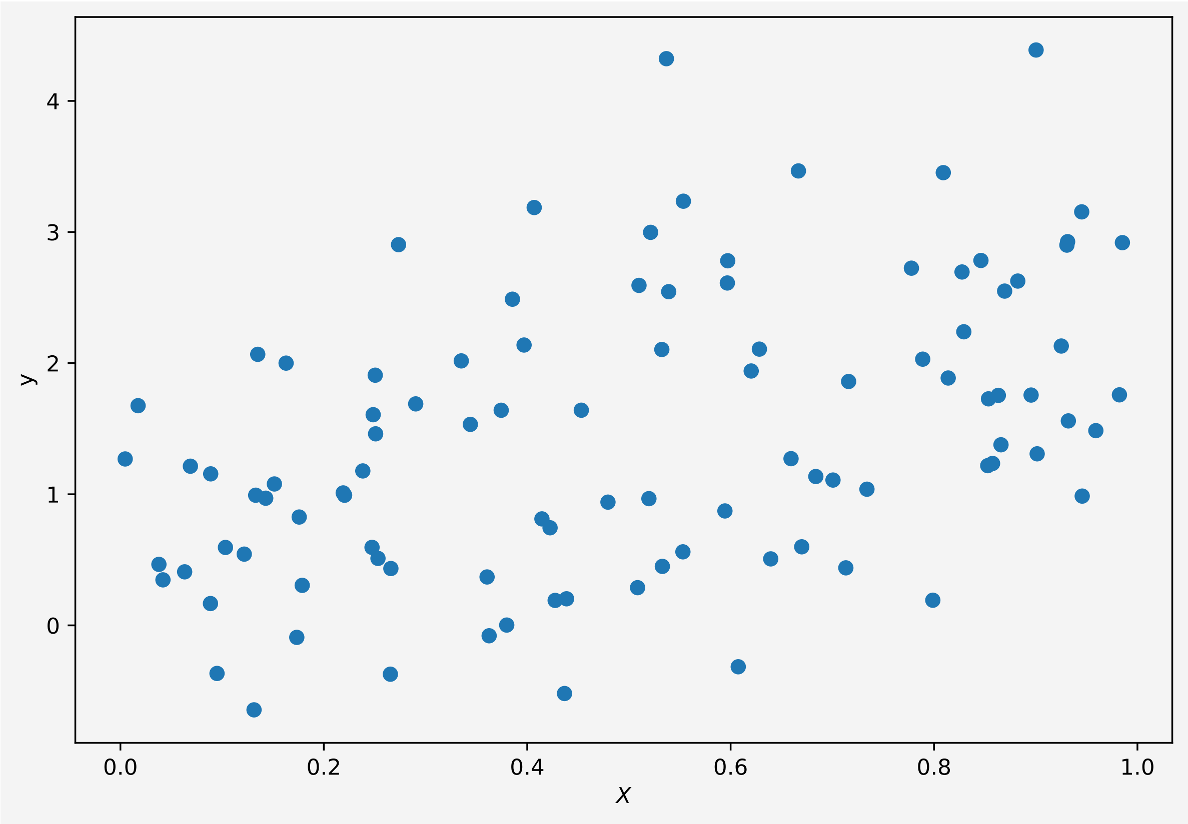
Synthetic Data

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.

```python
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.

```python
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([0.7340383 , 0.26593459, 0.84612546, 0.50990465, 0.55355   ,
       0.89544905, 0.22044372, 0.62022988, 0.33516663, 0.2383379 ,
       0.50850486, 0.93135315, 0.34407622, 0.4366772 , 0.25325133,
       0.92512659, 0.24853637, 0.21908213, 0.03785438, 0.53286609,
       0.04183943, 0.62828479, 0.25093675, 0.65939665, 0.1032779 ,
       0.39689971, 0.63942786, 0.36255945, 0.51965302, 0.90141699,
       0.94569894, 0.47938625, 0.1330192 , 0.29035671, 0.60749405,
       0.42245711, 0.45319168, 0.59439071, 0.37442281, 0.70065038,
       0.7777679 , 0.38546419, 0.4144583 , 0.66992251, 0.81394121,
       0.59676969, 0.98230375, 0.93208135, 0.17342988, 0.06315999,
       0.98522337, 0.40686237, 0.53910138, 0.59714902, 0.14286197,
       0.8694384 , 0.42746753, 0.71328543, 0.0885393 , 0.85357785,
       0.15139733, 0.12176088, 0.82929298, 0.09498705, 0.85752813,
       0.01737579, 0.53684054, 0.3798578 , 0.36052564, 0.882357  ,
       0.85282651, 0.86322943, 0.08877755, 0.90035328, 0.5323517 ,
       0.71602211, 0.2505533 , 0.2474065 , 0.55307544, 0.13138893,
       0.26545103, 0.2734753 , 0.79881515, 0.86584071, 0.95911284,
       0.17876019, 0.80910134, 0.94522694, 0.82752985, 0.17579447,
       0.683772  , 0.06880165, 0.66665655, 0.0046839 , 0.93064182,
       0.52131828, 0.78888237, 0.43865138, 0.13503125, 0.16291344])
```

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([[0.7340383 ],
       [0.26593459],
       [0.84612546],
       [0.50990465],
       [0.55355   ],
       [0.89544905],
       [0.22044372],
       [0.62022988],
       [0.33516663],
       [0.2383379 ]])
```

Now we fit the data to our model

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

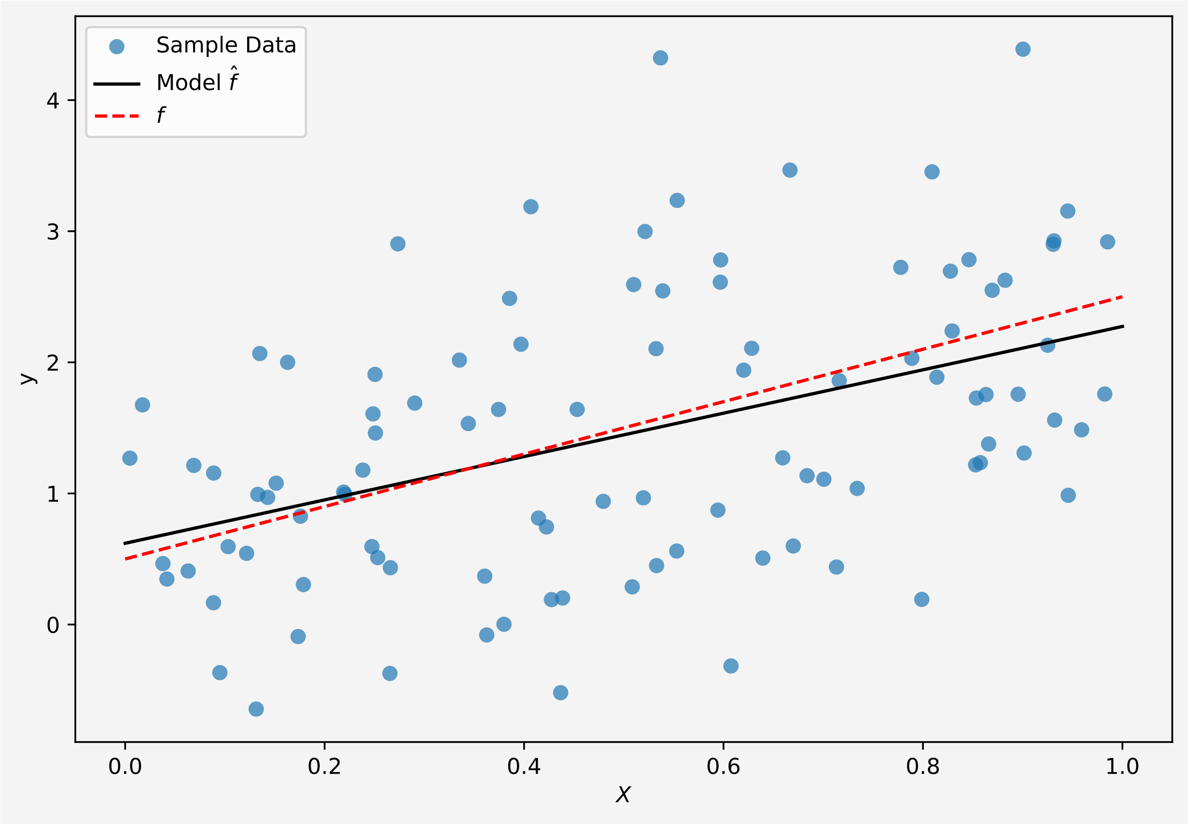
```
array([3.92726381, 5.58093586])
```

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 $\beta_0 = 0.6199$ and slope $\beta_1 = \text{array}([1.65367205])$.

```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