# 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 $\beta_0, \beta_1$. 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

$\beta_0, \beta_1$ we can minimize the error term as much as possible. So, we define the residual sum of squares (RSS) as:

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

Using multivariate calculus we see

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

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.

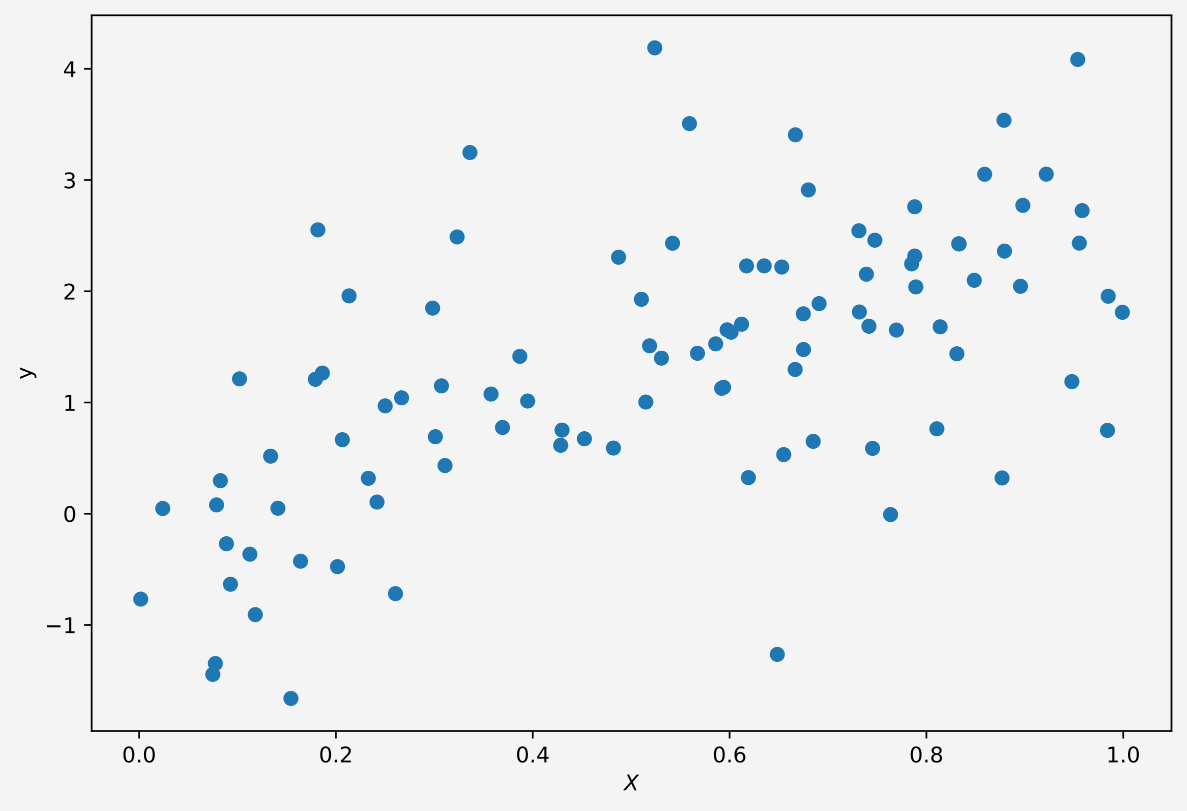
### Synthetic Data

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([0.17909053, 0.10204002, 0.65502301, 0.98399936, 0.66687881,
       0.45247821, 0.30720094, 0.56739152, 0.24174443, 0.11255791,
       0.36925249, 0.52399167, 0.13363344, 0.87687124, 0.59390243,
       0.78502843, 0.23290842, 0.95539584, 0.21333819, 0.92178666,
       0.3231584 , 0.18162161, 0.18620201, 0.48718241, 0.99916521,
       0.68004138, 0.95827014, 0.73143017, 0.09281234, 0.74529393,
       0.30103776, 0.35763128, 0.83278155, 0.59190062, 0.8980076 ,
       0.95380686, 0.74160128, 0.26666554, 0.5420169 , 0.61911437,
       0.73190229, 0.33615537, 0.61217397, 0.20160602, 0.81061291,
       0.61724709, 0.94782324, 0.98474521, 0.39480036, 0.78922069,
       0.48196753, 0.74765354, 0.42976208, 0.11807051, 0.07746642,
       0.69095973, 0.08873284, 0.8592124 , 0.78817248, 0.29825539,
       0.51040382, 0.73900621, 0.42837409, 0.514877  , 0.26048834,
       0.8139761 , 0.67508888, 0.83096608, 0.76357585, 0.78809893,
       0.15427795, 0.5858911 , 0.8793441 , 0.51871873, 0.31083898,
       0.59750894, 0.25004744, 0.60155242, 0.08258834, 0.84867079,
       0.14106236, 0.87886162, 0.83330256, 0.76951108, 0.68498518,
       0.67492983, 0.16402461, 0.89562741, 0.38683786, 0.02400288,
       0.63518135, 0.66663107, 0.07868824, 0.55921728, 0.20650629,
       0.07486824, 0.53075654, 0.00164944, 0.64849301, 0.65302329])
```

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([[0.17909053],
       [0.10204002],
       [0.65502301],
       [0.98399936],
       [0.66687881],
       [0.45247821],
       [0.30720094],
       [0.56739152],
       [0.24174443],
       [0.11255791]])
```

Now we fit the data to our model

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

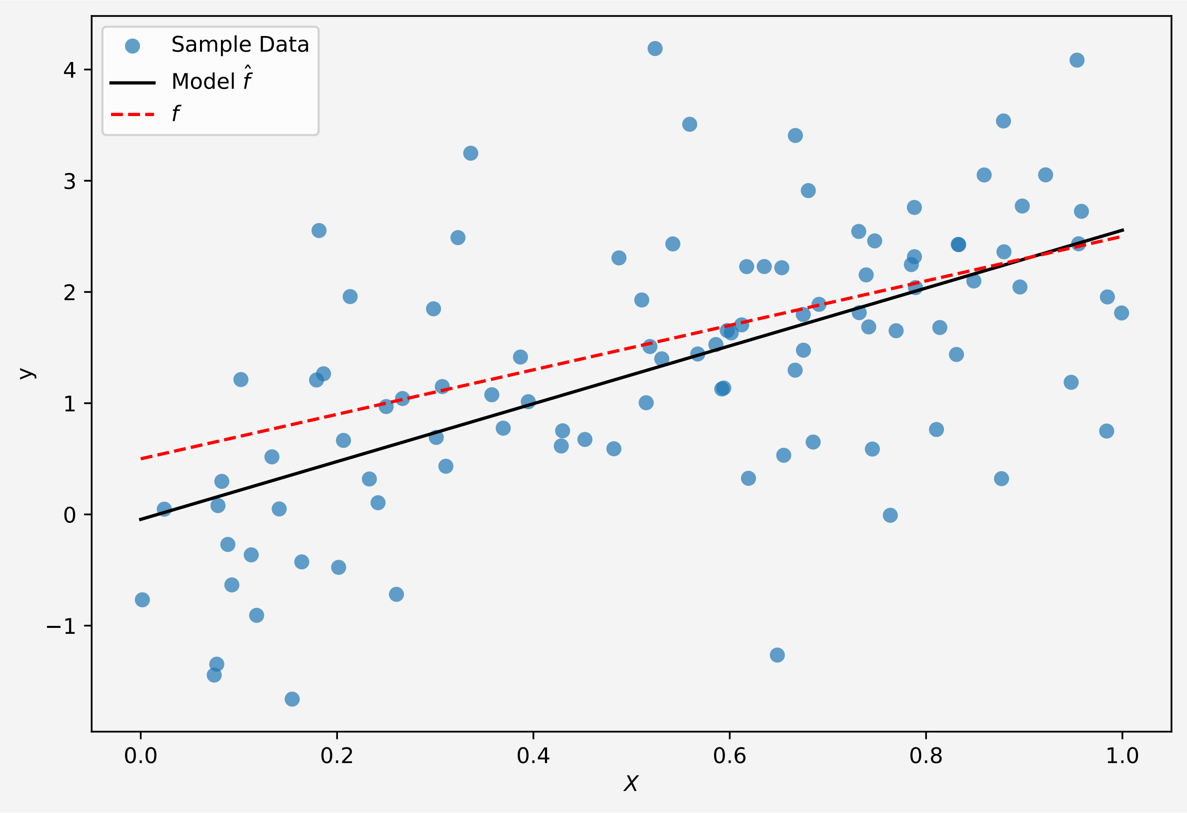
```
array([5.15521126, 7.75518389])
```

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 $\beta_0 =$ -0.0447 and slope $\beta_1 =$ array([2.59997263]).

```
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**