Simple Linear Regression from Scratch - Codealong

Introduction

In this codealong, you'll get some hands-on practice developing a simple linear regression model. In practice, you would typically use a code library rather than writing linear regression code from scratch, but this is an exercise designed to help you see what is happening "under the hood".

Objectives

You will be able to:

- Perform a linear regression using self-constructed functions
- Interpret the parameters of a simple linear regression model in relation to what they signify for specific data

Simple Linear Regression Recap

Remember that the ***data*** for a simple linear regression consists of y (the *dependent* variable) and x (the *independent* variable). Then the model ***parameters*** are the slope of the line, denoted as m or β_1, and the intercept (y value of the line when x is 0), denoted as c or β_0.

Thus the overall model notation is

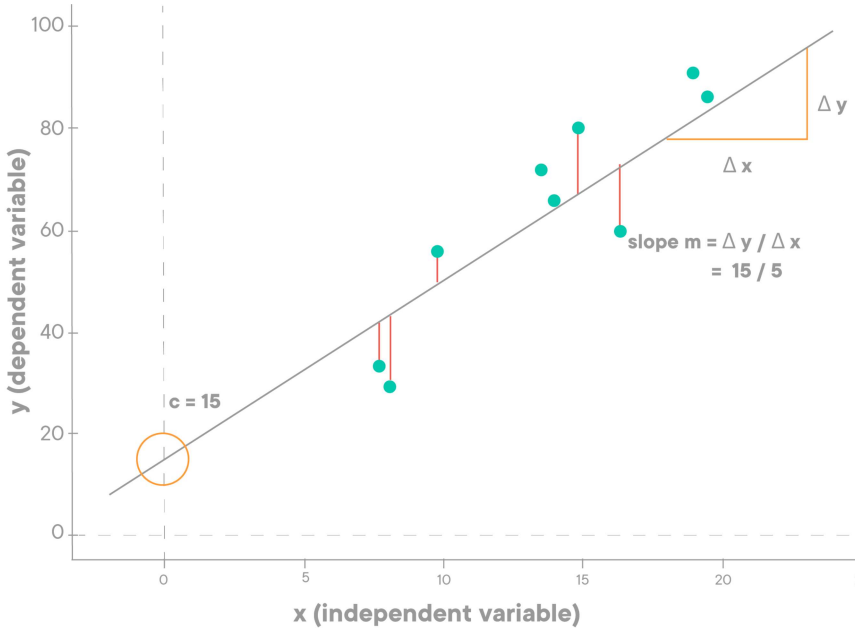
$$y = mx + c$$

or, alternatively

$$y = \beta_0 + \beta_1 x$$

In the example below, c is equal to 15 and m is equal to 3.

In other words, the overall equation is $y = 3x + 15$.



Finding Model Parameters

If you think back to the basic algebra formulas, you might remember that slope can be calculated between two points by finding the change in y over the change in x, i.e. $\Delta y/\Delta x$. But now you are dealing with messy data rather than perfect abstractions, so your regression line is not going to represent the relationship perfectly (i.e. there is going to be some amount of *error*). The question is how to find the ***best fit*** line, rather than just calculating $\Delta y/\Delta x$.

Because these are ***estimations***, we'll use the "hat" notation for the variables, i.e.

$$\hat{y} = \hat{m}x + \hat{c}$$

or

$$\hat{y} = \hat{\beta_0} + \hat{\beta_1} x$$

Everything in these equations represented with a "hat" (e.g. $\hat{y}$ rather than just y) means that it is an estimate or an approximation. The only part that is not using this notation is x, because we have the actual data values for the independent variable.

So, how do you find the line with the best fit? You may think that you have to try lots and lots of different lines to see which one fits best. Fortunately, this task is not as complicated as it may seem. Given some data points, **the best-fit line always has a distinct slope and y-intercept that can be calculated using simple linear algebraic approaches**.

The Least-Squares Method

We can calculate $\hat{m}$ (the slope of the best-fit line) using this formula:

$$\hat{m} = \rho\frac{S_y}{S_x}$$

Breaking down those components, we have:

- $\hat{m}$: the estimated slope
- ρ: the Pearson correlation, represented by the Greek letter "Rho"
- S_y: the standard deviation of the y values
- S_x: the standard deviation of the x values

(You can visit this Wikipedia link (https://en.wikipedia.org/wiki/Simple_linear_regression#Fitting_the_regression_line) to get take a look into the math behind the derivation of this formula.)

Then once we have the slope value ($\hat{m}$), we can put it back into our formula ($\hat{y} = \hat{m}x + \hat{c}$) to calculate the intercept. The idea is that

$$\bar{y} = \hat{c} + \hat{m}\bar{x}$$

so

$$\hat{c} = \bar{y} - \hat{m}\bar{x}$$

Breaking down those components, we have:

- $\hat{c}$: the estimated intercept
- $\bar{y}$: the mean of the y values
- $\hat{m}$: the estimated slope
- $\bar{x}$: the mean of the x values

Let's Get Started

In the cell below, we import the necessary libraries and provide you with some toy data:

In [ ]:
```
# Run this cell without changes

# import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')

# Initialize arrays X and Y with given values
# X = Independent Variable
X = np.array([1,2,3,4,5,6,8,8,9,10], dtype=np.float64)
# Y = Dependent Variable
Y = np.array([7,7,8,9,9,10,10,11,11,12], dtype=np.float64)
```

Scatter Plot

Before performing a linear regression analysis, it's a best practice to look at a scatter plot of the independent variable vs. the dependent variable. Linear regression is only appropriate if there is a linear relationship between them. In the cell below, create a quick scatter plot showing x vs. y.

Solution code (click to reveal)

In [ ]:
```
# Your code here
```

Based on the plot above, does linear regression analysis seem appropriate?

Answer (click to reveal)

In [ ]:
```
# Your answer here
```

Calculating the Slope

Write a function `calc_slope` that returns $\hat{m}$ for a given set of x and y data.

The formula is:

$$\hat{m} = \rho\frac{S_y}{S_x}$$

Remember that you can use NumPy methods to calculate correlation and standard deviation.

Solution code (click to reveal)

In [ ]:
```
def calc_slope(x_vals, y_vals):
    # Your code here

m = calc_slope(X,Y)
m # should produce approximately 0.539
```

Calculating the Intercept

Now that we have our estimated slope $\hat{m}$, we can calculate the estimated intercept $\hat{c}$.

As a reminder, the calculation for the best-fit line's y-intercept is:

$$\hat{c} = \bar{y} - \hat{m}\bar{x}$$

Write a function `calc_intercept` that returns $\hat{c}$ for a given $\hat{m}$, x, and y.

Solution code (click to reveal)

In [ ]:
```
def calc_intercept(m, x_vals, y_vals):
    # Your code here

c = calc_intercept(m, X, Y)
c # should produce approximately 6.38
```

Predicting a New Data Point

So, how might you go about actually making a prediction based on this model you just made?

Now that we have a working model with $\hat{m}$ and $\hat{c}$ as model parameters, we can fill in a value of x with these parameters to identify a corresponding value of $\hat{y}$ according to our model. Recall the formula:

$$\hat{y} = \hat{m}x + \hat{c}$$

Let's try to find a y prediction for a new value of $x = 7$.

Solution code (click to reveal)

In [ ]:
```
# Replace None with appropriate code
x_new = 7
y_predicted = None
y_predicted # should be about 10.155
```

Bringing It All Together

Write a function `best_fit` that takes in x and y values, calculates and prints the coefficient and intercept, and plots the original data points along with the best fit line. Be sure to reuse the functions we have already written!

Solution code (click to reveal)

In [ ]:
```
def best_fit(x_vals, y_vals):
    # Create a scatter plot of x vs. y

    # Calculate and print coefficient and intercept

    # Plot line created by coefficient and intercept

best_fit(X, Y)
```

So there we have it, our least squares regression line. This is the best fit line and does describe the data pretty well (still not perfect though).

Describe your Model Mathematically and in Words

What is the overall formula of the model you have created? How would you describe the slope and intercept, and what do they say about the relationship between x and y?

Answer (click to reveal)

In [ ]:
```
# Your answer here
```

Summary

In this lesson, you learned how to perform linear regression from scratch using NumPy methods. You first calculated the slope and intercept parameters of the regression line that best fit the data. You then used the regression line parameters to predict the value ($\hat{y}$-value) of a previously unseen feature (x-value). You finally plotted your model against the original dataset.