Linear Regression From Scratch

Ruhani Rekhi

1 Introduction

The goal of this project is to implement linear regression from scratch using Python to understand the underlying mathematics and algorithms behind linear regression. The importance of the project lies in its ability to provide a foundational understanding of linear regression as well as insights into how the algorithm works, including the concepts of cost functions, gradient descent, and model evaluation.

1.1 Basics Of Linear Regression

Regression is a method used to model the relationship between a dependent variable and one or multiple independent variables. Linear regression is a specific type of regression analysis where we model the relationsip between the dependent variable and independent variable(s) as a linear function. The main goal of linear regression is to determine the function of a best fit line that minimizes error. Error in this case is defined as difference between the predicted and actual value. Mathematically written as $e = y - \hat{y}$ where $\hat{y}$ is the predicted value. Our function is defined as $\hat{y} = b_0 + b_1 x_1$ where b_0 is our y-intercept and b_1 is the slope.

1.2 Minimizing the Error Function

To begin minimizing the error function we need to define what the error function is.

Let E be our error function defined as :

$$E = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y_i})^2 = \frac{1}{n}\sum_{i}^{n}(y_i - (b_0 + b_1 x_1))^2$$

This value is referred to as the MSE or mean squared error. In layman terms, we are taking the difference from all the actual points y-value and what our linear function would predict what the y-value would be and square the difference and divide it by the total number of points to derive our MSE. Now that we have defined our error function E we now want to minimize our E : find the function with lowest possible E. The only things we can control are our b_0 (y-intercept) and b_1 (slope) since our x and y are just our data points. Simply put, we want to find the slope and intercept that minimize E. We can do this by taking the partial derivative of E in respect to b_0 and b_1. This gives us the direction of the steepest ascent with respect to b_0 and b_1. The method of steepest ascent maximizes a function by moving direction of the gradient thus we would be finding a b_0 and b_1 that would maximally increase E. This would help in minimizing E since if we know the direction of the steepest ascent we can go the opposite direction and find the steepest descent.

$$\begin{aligned}\frac{\partial E}{\partial b_1} &= \frac{1}{n}\sum_{i=0}^{n}(2(y_i - (b_0 + b_1 x_1)) * (-x_i)) \\ &= \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - (b_0 + b_1 x_1))\end{aligned} \tag{1}$$

$$\frac{\partial E}{\partial b_0} = \frac{-2}{n}\sum_{i=0}^{n}(y_i - (b_0 + b_1 x_1)) \tag{2}$$

We have the direction of the steepest ascent in respect to b_0 and b_1, so now we have to go the opposite direction. To do this for each iteration we take our new b_1 and b_0 and assign the current b_1 and b_1 subtracted by our learning rate multiplied by the direction of the steepest ascent

$$\begin{aligned} b_1 &= b_1 - L * \frac{\partial E}{\partial b_1} \\ b_0 &= b_0 - L * \frac{\partial E}{\partial b_0} \end{aligned} \tag{3}$$

This approach helps us accomodate many variables when working with linear regression. We are subtracting $\frac{\partial E}{\partial b_1}$ and $\frac{\partial E}{\partial b_0}$ from the respective variables to go the opposite direction of the steepest ascent. The learning rate determines how large the steps we take in that said direction. The larger our L (learning rate) the faster we can get to optimization, the lower the rate it will take longer to reach optimization but we can pay attention to more details. We will try to utilize a learning rate of around 0.0001. Now we will apply the theory explained in this paper to Python!