Daniel Kwon

1. Problem description

The main challenge of this assignment was to design a program that will train the data with multi-variant linear regression model using the stochastic gradient descent algorithm. We were given two data sets to test our algorithm on, one with 8 variables in its dimension (Concrete_Data.xls) and one with just one (hour_score.txt).

The first task was to devise a code to read the data from both of the datasets and clean it into a usable dataframe. My choice of language to do this task was Python (version 3.9.2) since the program is easy write and replicate with vast amount of helpful libraries.

2. Major steps

The objective function of LSE used for this task was the following:

$$L(w) = \frac{1}{2}\sum_{k=1}^{n}(t_k - w * d_k)^2$$

with no particular activation function since we are dealing with linear regression. Since we are aiming to locate the local minimum of the function above, we work with the gradient of out objective function, which is:

$$\nabla w_k = \sum\nolimits_{d \in D}(y_d - t_d)d(k)$$

Where d(k) is the k-th element of the example being observed in the training.

Since we have the gradient ready, now we proceed to implementing the algorithm for the training process. By this point, we've cleaned our data and separated the y values from the rest of independent variables and have padded each vector element of the x values with 1 as the 0th dimension.

3. Update rule

In order to maximize the detailed learning, the learning coefficient for this project was selected to be 0.001 * gradient. Each of the variables in the model is separately updated after the gradient is calculated with appropriate arguments.

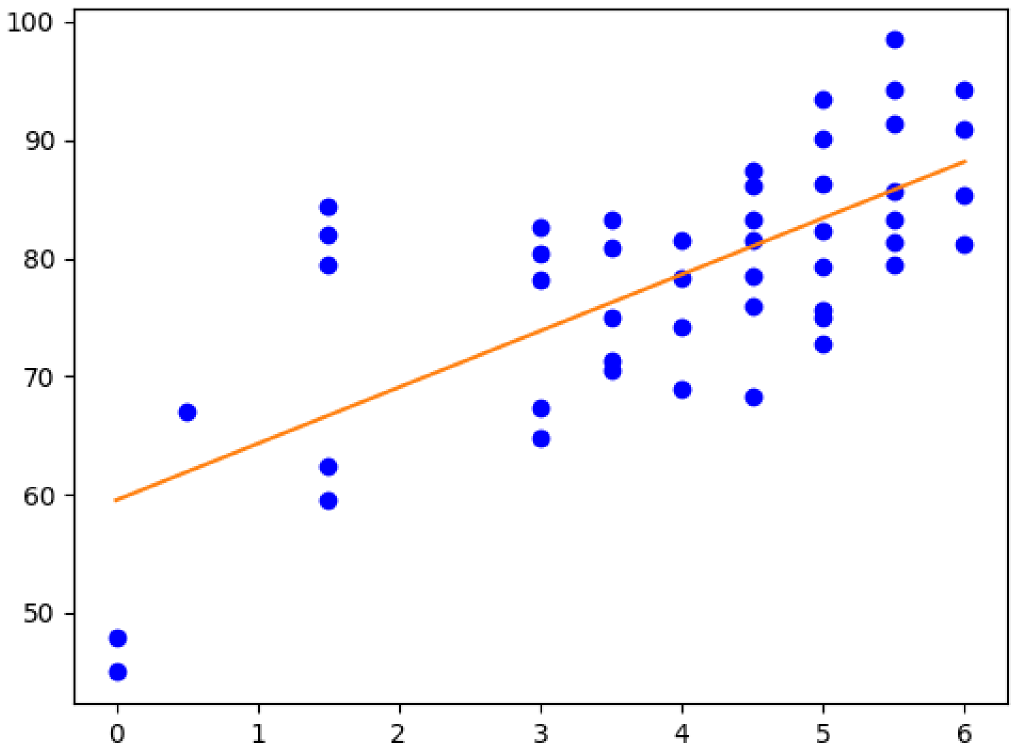
4. Pseudocode

For the gradient descent algorithm's implementation, we use the following pseudocode:

```
Gradient_descent (regression, x, y, iterations):
    While (I < iterations):
        For (k in range(regression.length):
            For d in range(len(x)):
                Sum += (y(d) – t(d))*d(k)
            Regression[k] -= 0.001*sum
        I+=1
        If (each step < threshold):
            break
```

This way, we iterate through the given number of iterations to learn the dataset, and come to a stop when we face a situation where each change is less than the given threshold.

5. Results



The image above is the linear regression graph plotted through the matplotlib library for the hour_score.txt dataset.