Assignment Problem: Gradient Descent Algorithm in Python

Consider the following 3 datapoints.

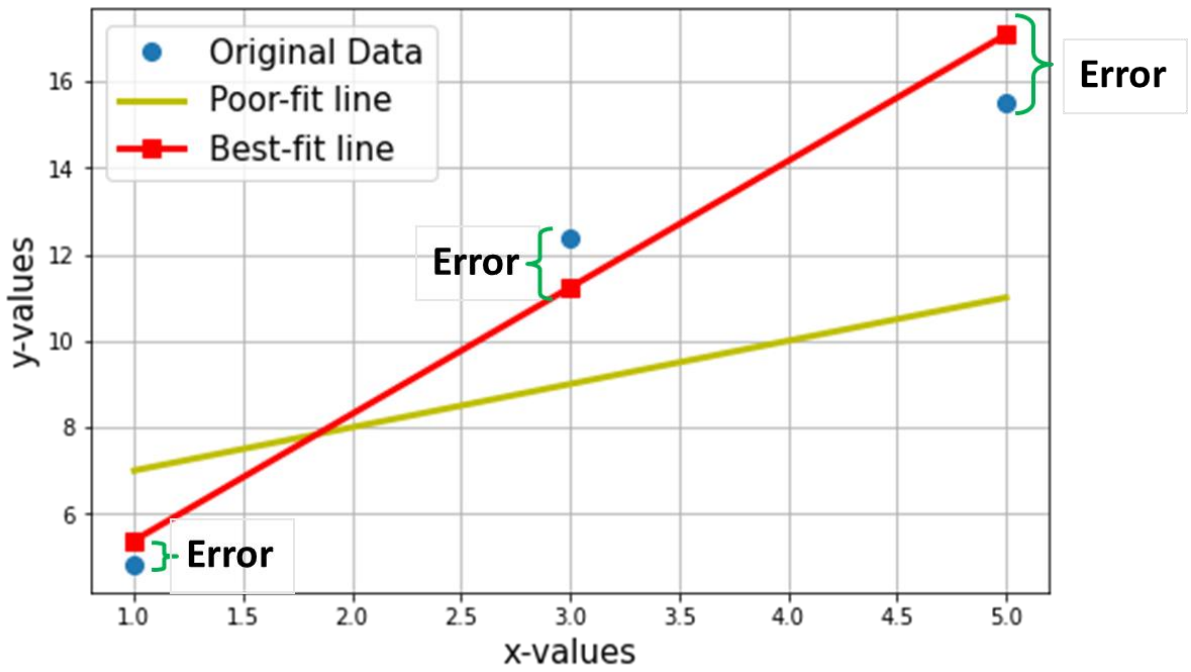
X1 (Feature)	Y (Target)
1	4.8
3	12.4
5	15.5

- The following plot shows these 3 datapoints in Blue circles. Also shown is the red-line (with squares), which we are claiming is the **"best-fit line".**

- The claim is that this best fit-line will have the minimum error for prediction (the predicted values are actually the red-squares, hence the vertical difference is the error in prediction).

- This total difference (error) across all the datapoints is expressed as the Mean Squared Error Function, which will be minimized using the Gradient Descent Algorithm, discussed below.

- Minimizing or maximizing any quantity is mathematically referred as an Optimization Problem, and hence the solution (the point where the minima/maxima exists) is referred the **"optimal values"**.



Here X1 refers to the independent variable (also called as Feature / Attribute in Machine Learning)

Y is the dependent variable (also known as Target Variable in ML)

The net Objective is to find the Equation of the **Best-Fitting Straight Line** (through these 3 data points, mentioned in the above table, also represented by the blue circles in the above plot).

$$\hat{\mathbf{Y}} = \mathbf{w}_0 + \mathbf{w}_1\mathbf{X}_1 \qquad \text{---------- (1)}$$

is the equation of the best-fit line (red-line in the plot)

w_1 = slope of the line; w_0 = intercept of the line

w_0 and w_1 are also called **model weights.**

Y^ is the predicted values of Y, given by the "best-fit line". These predicted values are represented by red-squares on the red-line. Of course, the predicted values are NOT exactly same as the actual values of Y (blue circles), the vertical difference represents the error in the prediction given by:

$$\mathbf{Error_i} = \hat{\mathbf{Y}}_\mathbf{i} - \mathbf{Y_i} \qquad \text{---------- (2)}$$

for any i^{th} data point.

$$\mathbf{MSE} = \frac{1}{\mathbf{N}}\sum_{\mathbf{i}=1}^{\mathbf{N}}(\mathbf{Error_i})^2 = \frac{1}{\mathbf{N}}\sum_{\mathbf{i}=1}^{\mathbf{N}}(\hat{\mathbf{Y}}_\mathbf{i} - \mathbf{Y_i})^2 \qquad \text{---------- (3)}$$

N = Total no. of data points (in this case, it is 3)

Now the problem is to find the **"optimal values"** of the slope and intercept of this best-fit line, such that the "Mean Squared Error" (MSE) is minimum. You can easily see that the yellow-line (a poor-fit line) which has "non-optimal" values of slope & intercept fits the data very badly (btw the exact equation of the yellow line is **x+6**, so slope is 1 and intercept is 6 units)

How will i get the optimal values of the slope and intercept ????

This is where the Gradient Descent Algorithm comes!

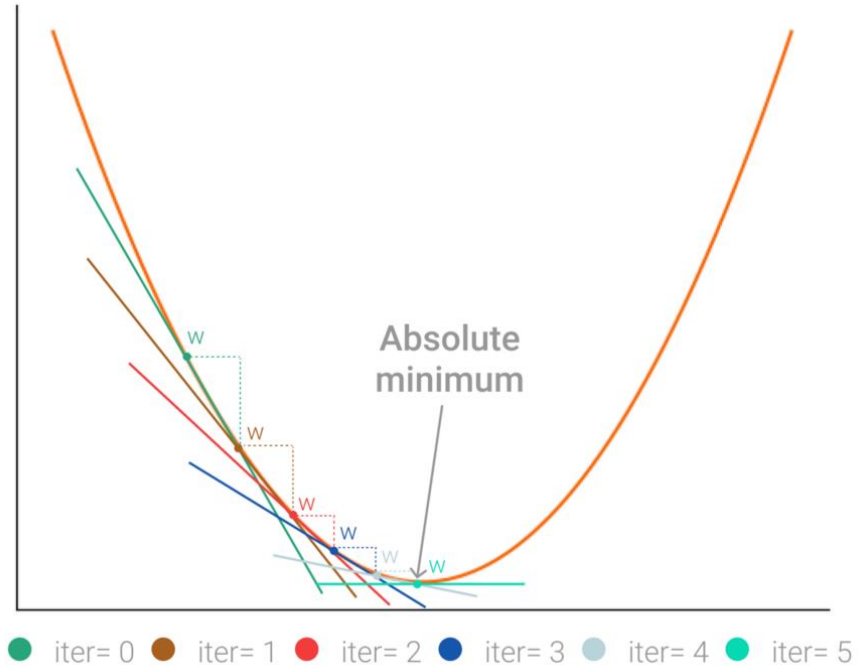
$$w_0^{k+1} = w_0^k - \left(\alpha \sum_{i=1}^{N} (\hat{Y}_i - Y_i) \right)$$

$$w_1^{k+1} = w_1^k - \left(\alpha \sum_{i=1}^{N} [(\hat{Y}_i - Y_i) \times X_{1i}] \right) \qquad \text{---------- (4 \& 5)}$$

where w_0^k & w_1^k represent the values of the intercept and the slope of the linear-fit in the k^{th} iteration, whereas w_0^{k+1} & w_1^{k+1} represent the values of the intercept and the slope of the linear-fit in the $(k+1)^{th}$ iteration (next iteration). **w0 & w1 are also called as model weights or model coefficients.**

α represents the Learning Rate.

The derivation of the above equations will be done in the machine learning lessons.



Gradient Descent Algorithm

1. Initialize the algorithm with random values of α, and weights (w_0, w_1)
2. Calculate predictions Y^ = w_0 + w_1 * x as shown in the equation 1
3. Calculate Error terms & MSE Loss Function (L):

   Error terms are: $\sum_{i=1}^{N}(\hat{Y}_i - Y_i)$ and $\sum_{i=1}^{N}[(\hat{Y}_i - Y_i) \times X_{1i}]$ for the datapoints i=1 to N (here N is equal to 3)

   Loss function as shown in equation 3

4. **Update your weights:** Using equations 4 and 5
5. Repeat 2-4, until convergence.

Based on the above-mentioned steps, we can calculate the weights. Let the learning rate (α) be 0.01 and initialize the weights w_0 and w_1 as 0.

X1	Y	Y^	Y^-Y	(Y^-Y)*X1
1	4.8	0	-4.8	-4.8
3	12.4	0	-12.4	-37.2
5	15.5	0	-15.5	-77.5
		Sum	-32.7	-119.5

$w_0^1 = 0 - (0.01) * (-32.7) = 0.327$

$w_1^1 = 0 - (0.01) * (-119.5) = 1.195$

With these new weights, let us update the above table.

X1	Y	Y^	Y^-Y	(Y^-Y)*X1
1	4.8	1.522	-3.278	-3.278
3	12.4	3.912	-8.488	-25.464
5	15.5	6.302	-9.198	-45.99
		Sum	-20.964	-74.732

$w_0^2 = 0.327 - (0.01) * (-20.964) = 0.537$

$w_1^2 = 1.195 - (0.01) * (-74.732) = 1.942$

With these new weights, let us update the above table.

X1	Y	Y^	Y^-Y	(Y^-Y)*X1
1	4.8	2.479	-2.321	-2.321
3	12.4	6.363	-6.037	-18.111
5	15.5	10.247	-5.253	-26.265
		Sum	-13.611	-46.697

As we can see, the sum of errors is decreasing as we are updating the weights.

We can continue to update the weights like the above manner until the sum of errors become minimum (i.e. reaches almost a constant value)

Finally I want the plots of :

- Loss function (y-axis) vs w0 (x-axis)
- Loss function (y-axis) vs w0 (y-axis)
- 3D-plot of Loss function w.r.t. w0 & w1 similar to below image