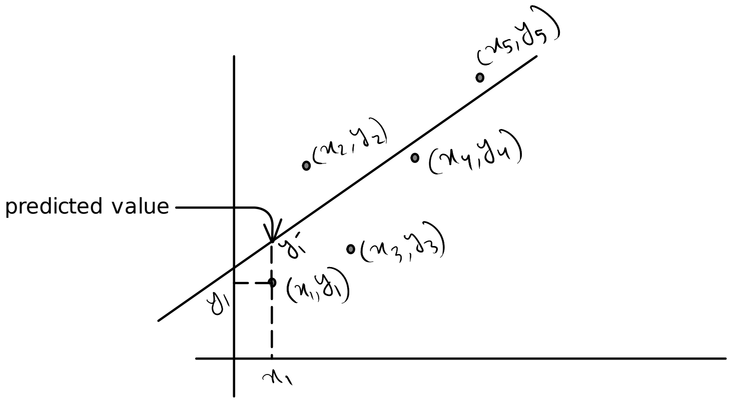
Linear Regression and Gradient Descent



Here, the erorr is $y_1' - y_1 = -(y_1 - y_1')$ for the first point and $y_2 - y_2'$ for the second point

We can take square of each error to get the positive value.
The error function will take the whole line as input and outputs the sum of all errors at each input point.

A line can be modelled only by slope (m) and bias (c) value so the input parameter of the error function is m and c

Therefore, the error function or loss function or the cost function is

$$erf(m,c) = (y_1 - y_1')^2 + (y_2 - y_2')^2 + (y_3 - y_3')^2 + \dots$$
$$= \sum_{i=1}^{n} (y_i - y_i')^2$$
$$= \sum_{i=1}^{n} \{y_i - (mx_i + c)\}^2$$
$$= \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

This function is called sum of square residual (SSR), there are more error function other than this.
Now, we have to choose the value of m and c such that erf(m,c) can be lowest as possible
We can apply gradient descent for finding that m and c

Steps of Gradient Descent:

The error function will be maximum or minimum with respect to m and c when

$\frac{\partial}{\partial m} erf(m,c) = 0$ And $\frac{\partial}{\partial c} erf(m,c) = 0$

Note, that the error function is always like a upward cup if you plot it. So, the function must be minimum when both partial derivatives are zero, you can also test yourself by second derivative test.
So, the goal of SDG is to satisfy these equations.

This algorithm takes an initial guess of m and c and iteratively goes towards a value for which the Gradient is zero.
Now, lets say, we choose an initial value for m and c, now we find the gradient with respect to those values. if we get a negative gradient with respect to m that means we have to increase the value of m and we have to decrease otherwise to get closer to the 0 gradient, same goes for c . Think of a upward cup shaped parabola if you choose a point on the parabola that is left side of the minimum value you have to increase that vlaue to go toward the minimum value or decrease otherwise. you can also check the second differentiation of the error function with respect to m or c that is positive and this tells that the gradient will increase if m or c increase. so if you have negative gradient you will have to increase m or c to increase the gradient so that you can get closer to the zero gradient. On the other hand, if you get positive gradient you have to decrease the value of m or c to decrease the gradient to get towards zero gradient. So you can set the step size proportional gradient value and set m' = m-stepsize or c' = c-stepsize where m' and c' are new value. The proportionality constant here is called learning rate.

So, step size = gradient value * learning rate
and new value m' = m - stepsize and c' = c-stepsize
We can get the gradient value by partially differentiating error function with respect to m or c

$$
\begin{aligned}
&\frac{\partial}{\partial m} erf(m,c) \\
&= \frac{\partial}{\partial m} \sum_{i=1}^{n} (y_i - mx_i - c)^2 \\
&= \sum_{i=1}^{n} 2(y_i - mx_i - c) \frac{\partial}{\partial m} (y_i - mx_i - c) \\
&= 2 \sum_{i=1}^{n} (y_i - mx_i - c)(-x_i)
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial}{\partial c} erf(m,c) \\
&= \frac{\partial}{\partial c} \sum_{i=1}^{n} (y_i - mx_i - c)^2 \\
&= \sum_{i=1}^{n} 2(y_i - mx_i - c) \frac{\partial}{\partial c} (y_i - mx_i - c) \\
&= 2 \sum_{i=1}^{n} (y_i - mx_i - c)(-1)
\end{aligned}
$$

We have to do multiple iterations of this steps to get the gradient close to zero.

Here is an example of one iteration with 3 points (1, 2), (2, 3), (-1, 2) and learning rate = 0.5

Now, let's take m = 1, and c = 2 initially.

Therefore,

$$
\begin{aligned}
&\frac{\partial}{\partial m} erf(m,c) \\
&= 2 \sum_{i=1}^{3} (y_i - mx_i - c)(-x_i) \\
&= 2 \sum_{i=1}^{3} (y_i - 1x_i - 2)(-x_i) \\
&= 2(y_1 - x_1 - 2)(-x_1) + 2(y_2 - x_2 - 2)(-x_2) + 2(y_3 - x_3 - 2)(-x_3) \\
&= 2(2 - 1 - 2)(-1) + 2(3 - 2 - 2)(-2) + 2(2 + 1 - 2)(1) \\
&= 8
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial}{\partial c} erf(m,c) \\
&= 2 \sum_{i=1}^{3} (y_i - mx_i - c)(-1) \\
&= 2 \sum_{i=1}^{3} (y_i - 1x_i - 2)(-1) \\
&= 2(y_1 - x_1 - 2)(-1) + 2(y_2 - x_2 - 2)(-1) + 2(y_3 - x_3 - 2)(-1) \\
&= 2(2 - 1 - 2)(-1) + 2(3 - 2 - 2)(-1) + 2(2 + 1 - 2)(-1) \\
&= 2
\end{aligned}
$$

Therefore, our step size for m will be 8*0.5 = 4 and step size for c will be 2*0.5 = 1.
So, the new updated value of m and c will be,
m' = m-stepsize of m
=> m' = m-4 = 1-4 = -3 and
c' = c-stepsize of c
=> c' = c-1 = 2-1 = 1
Now, we have to continue few more iterations like this to get better value.