Gradient Descent (In simple words!)

The machine is taught, and it learns. When tested, it makes some mistakes. To overcome its mistakes, the machine needs another tutor. Gradient Descent is that tutor.

Gradient is another word for derivative of a function and descent means stepping ‘down’.

The derivative of any function (special relation between two variables) shows the ‘rate’ (speed) of change in the outputs of that function with respect to corresponding inputs.

Gradient Descent involves computing the derivative of the machine learning model’s (including neural networks) function and using that value to adjust the weights of inputs or instances.

The weights are adjusted so that the loss or error gets minimized i.e. the models starts overcoming its mistakes and learns more accurately.

We seek accuracy! Meaning, we want the error value to be zero. To do that, we have to minimize/reduce the error value.

Now that the model knows how to overcome its mistakes (as it has calculated the gradient of the error or loss function), it has to adjust the weights of the inputs appropriately. It can use trial and error! But since its intelligent, it chooses patience and takes one step at a time. These steps are called Learning Rate.

Following is a simple demonstration of the working of the Gradient Descent algorithm:

> One of the (thousand) instances has actual (original, as recorded in real environments) value say 77 and its weight is x.
>
> We chose a machine learning model (consider Linear Regression model) and started training it on this dataset. After training, we will tell it to use the same data and try predicting the output using available inputs and let’s say for the above-mentioned instance with actual value 77, the model predicts a value 60.
>
> And this difference in actual and predicted value is called error, which the model has to overcome to be completely accurate.
>
> We have the difference of 17 here. The Gradient Descent algorithm will first calculate gradient of the error function.
>
> The equation of weight update is:
>
> First update of weight (of a specific feature) = Previous weight – (Learning Rate * Gradient)
>
> As mentioned above, the previous or initial weight of the feature is ‘x’.
>
> Learning Rate controls the speed (gradient) of the learning (or correction in learning).
>
> The weight update step is repeated until the gradient value reaches zero. It means, the model has minimized the error.

And that’s how it works!