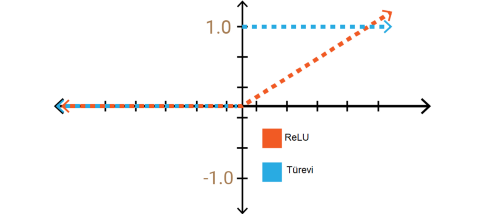
How Relu brings nonlinearity ?



ReLU **is** a non-linear function, there is no way you could get any shapes on the graph having only linear terms, any linear function can be simplified to a form **y** = **ab + x**, which is a **straight line.**

The simple answer is that Relu output is not a straight line, it bends at the x-axis. The more interesting point is what’s the consequence of this non-linearity. In simple terms, linear functions allow you to dissect the feature plane using a straight line. But with the non-linearity of Relus, you can build arbitrary shaped curves on the feature plane.

How Neural Networks can learn to square an input ?

X = [[10, 88, 14 ,2 ,98, 8, 32, …, 144]] #(Data)

Y = [[100, 7744, 196, 4, 9604, 64, 1024, …, 20736]] #(Label)

X.shape = (1, nx) , Y.shape = (1, ny)

W = np.random.randn((1, nx)) #(Weight)

Y’ = X \* W.T , Loss = Y - Y’ #(Linear function)

W = W - α \* dW #(Parameter update)

1-) In order for Artificial Neural networks to learn to square their inputs, we have to create a training data and create tag data consisting of squares of this training data.

2-) We must create a parameter that can learn from this data. The reason we added the bayes vector is that we no longer need a number like + C when calculating (X ^^ 2).

3-) Later, I created a linear function Y '= X \* W.T and got the estimates, namely Y'.

4-) I have given these predictions and tags to a loss function and update my W parameter according to the value returned from the loss function.

5-) And so the W parameter in the form of forward and backward propagation will perform the learning process.