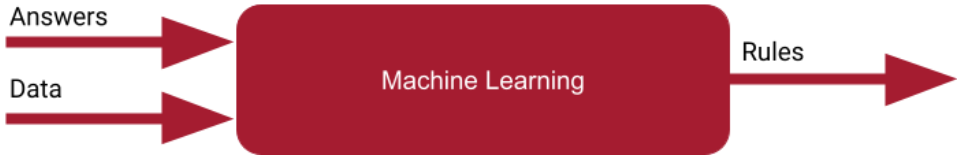
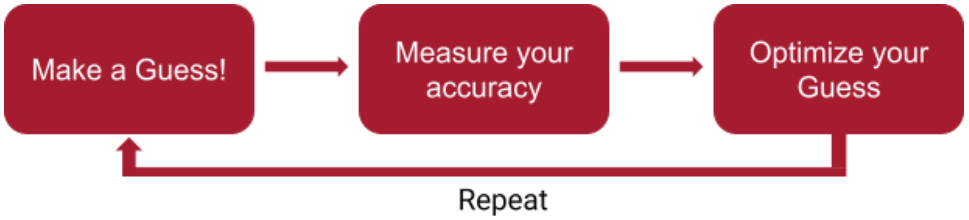
Initialization in Machine Learning

Recall that Machine Learning, the core process powering TinyML, is the process of having Data and Answers, and from them attempting to infer the rules that link them:



So, if your **data** is the numbers in this set: [-1, 0 , 1, 2, 3, 4] , and your corresponding **answers** are the numbers in this set: [-3, -1, 1, 3, 5, 7], one way to begin to infer the relationship between them (assuming it's linear) is to have a function, where we say Y = wX+b, and we have to try to figure out the values of w and b.

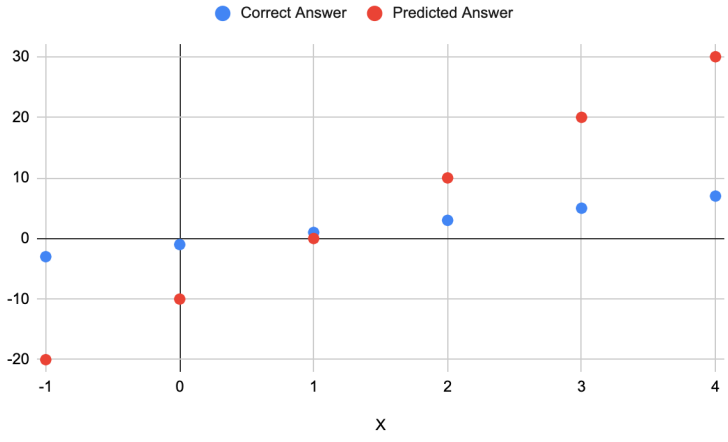
A process to do this is shown below:



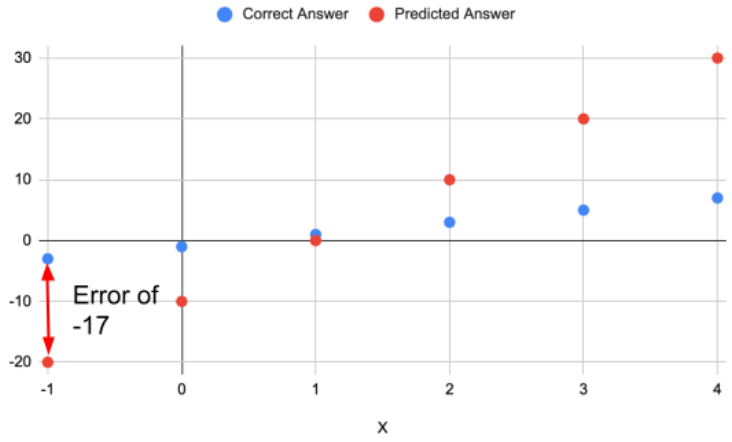
We could make a guess as to the values of w and b, and measure how accurate that guess is.

So, for example, if we guess that w=10, and b = -10, we would get the set of **answers** for our data as [-20, -10, 0, 10, 20, 30], where our *correct* answers are [-3, -1, 1, 3, 5, 7].

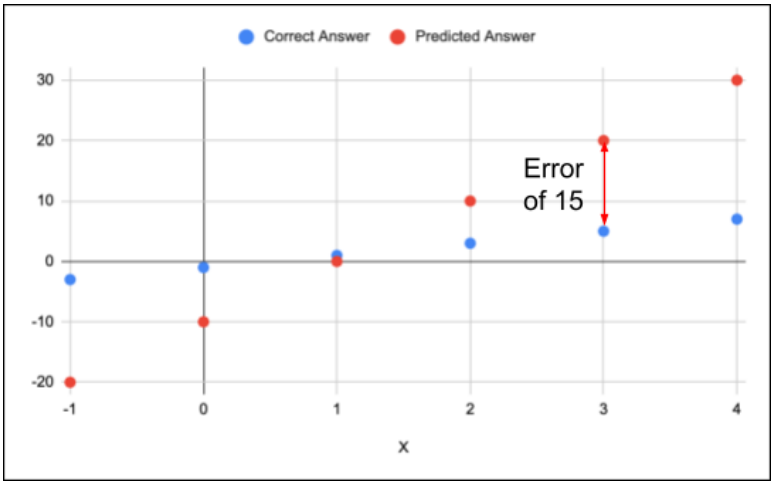
From this we can measure our *loss*, by plotting our predicted answer against our actual answer.



So, for example, for x=-1, we predicted -20, when the answer is actually -3, so we are off by -17



And when x=3, we predicted the answer to be 20, when in fact it is supposed to be 5, so we have an error of 15.

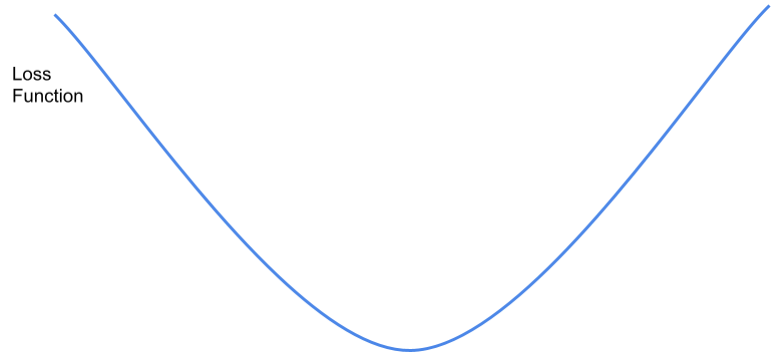


If we want to find out our error, we could average each of these.

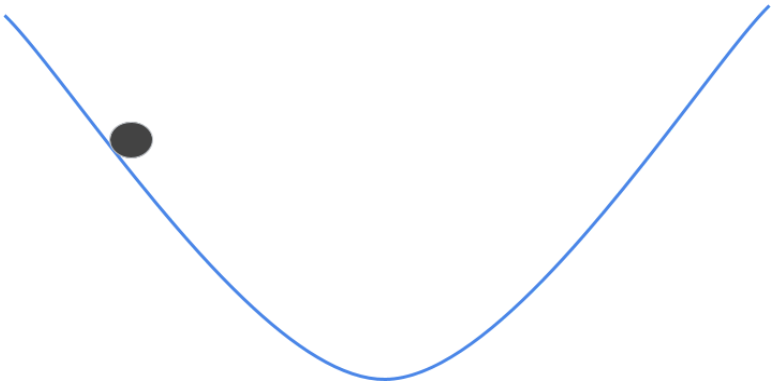
However, by doing this our negative (-17) and positive (15) errors would mostly cancel out, so the smart thing to do to ensure that we don't have them do this is to *square* the error amount, average out the squares of the errors, and then get the square root of that.

This gives us a loss *function* of root mean squared error.

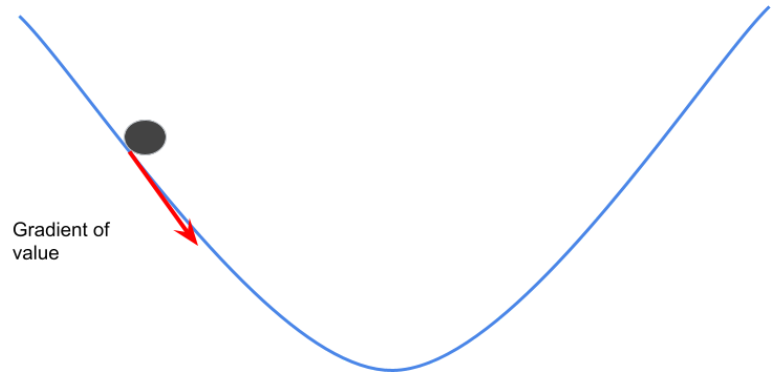
When we now want to optimize our guess, and do better than these results, we can look at the loss function, and figure out a way to minimize our loss. As the loss function uses a *square*, the curve of the function is parabolic.



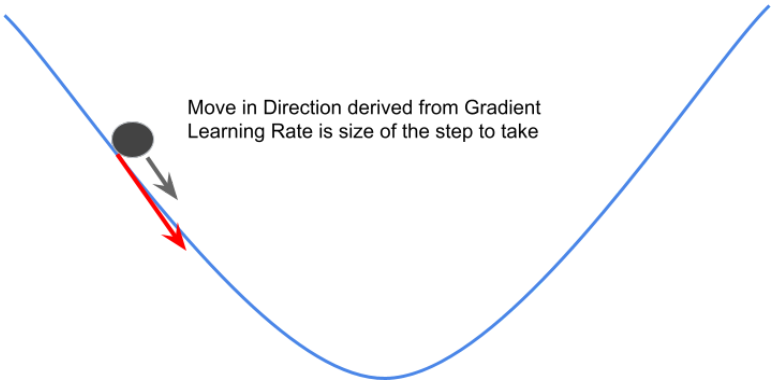
And the value of our loss with our current parameters can be plotted on this. Our loss was high, so we can say we're pretty far up the curve.



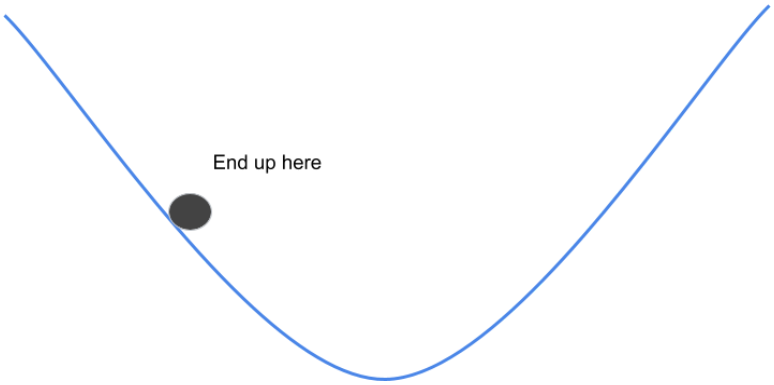
When we differentiate our values against the loss function, we'll get a gradient, and we can use the gradient to figure out which direction we have to go in to move down the slope!



And we can then move down the slope in the direction derived from gradient. We'll 'jump' by a certain amount, and we can call that amount the 'Learning Rate'



After that, we are now closer to the bottom of the curve. The parameters that give us our location on the curve can then be the next ‘guess’, and by definition, these will have a lower loss, and we can repeat the loop. This is called *back propagation*.



The process can be repeated, and over time the value will get closer and closer to the bottom of the parabola, which is the minimum value of the loss function.

This process of using the gradient of the value to figure out the direction of the minimum, and then jumping down the curve towards the minimum is called *gradient descent.*