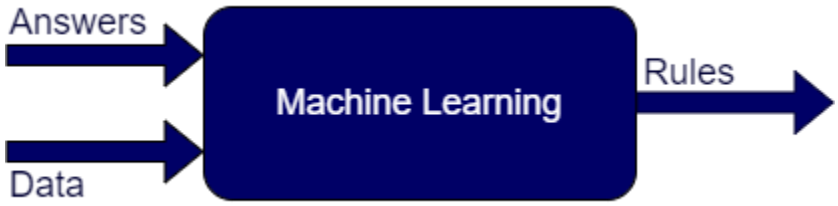
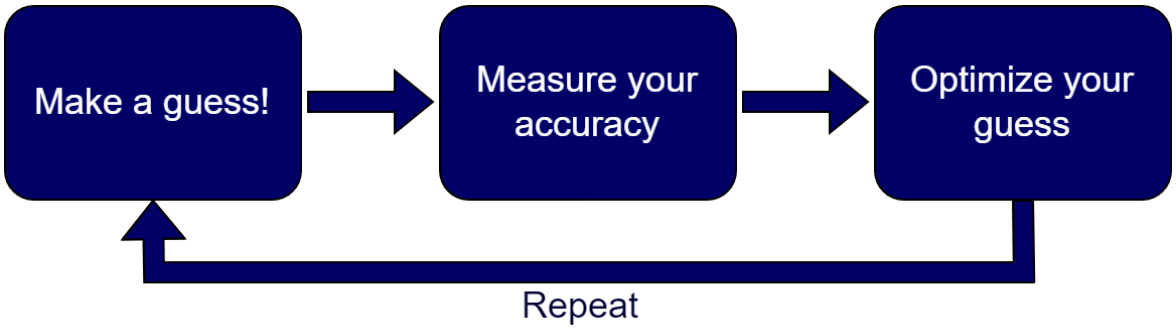
Initialization and Learning

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 consists of 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 inferring the relationship between them (assuming it's linear) is to consider a function, Y = wX + b, and try to figure out the values of w and b.

A process to do this is shown below:

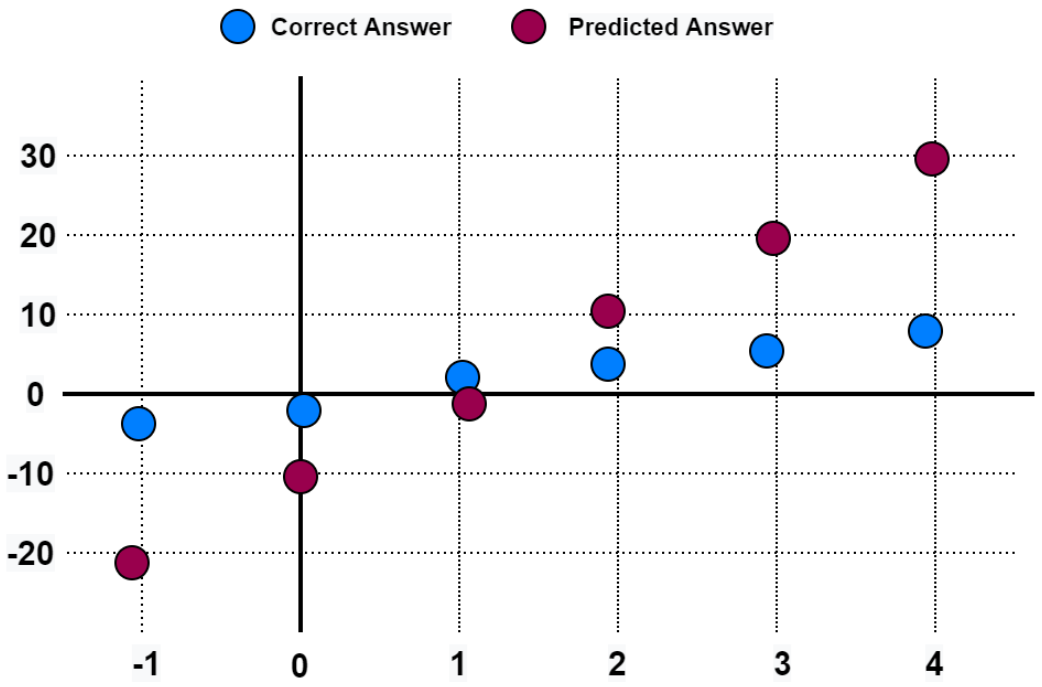


We could make a guess as to the values of w and b, and measure how accurate that guess is.

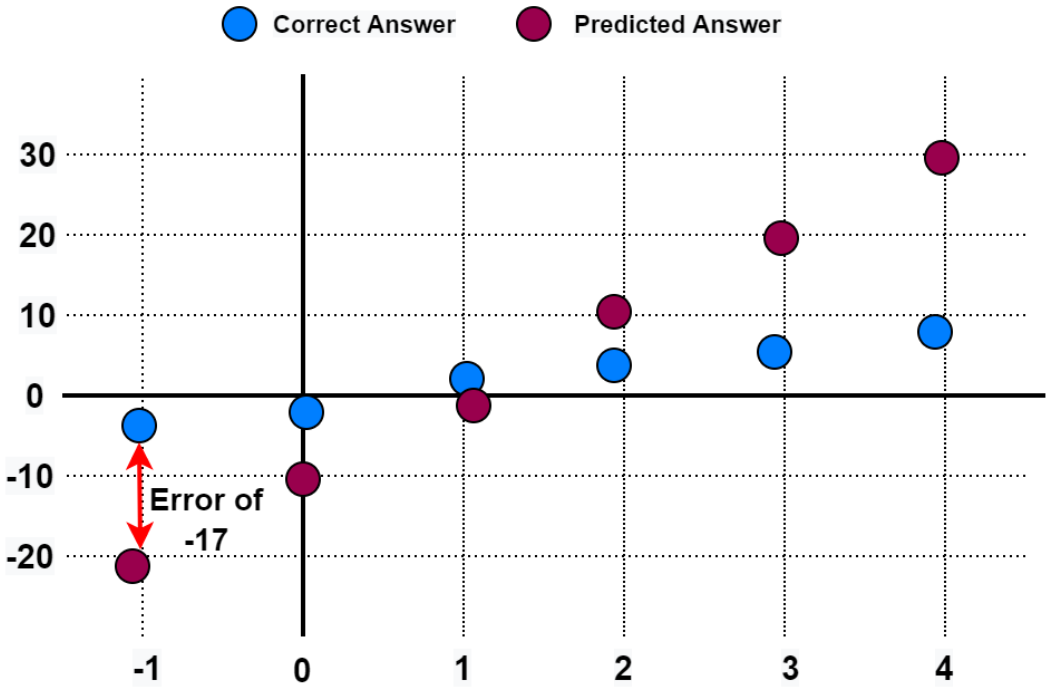
So, for example, if we guess that w = 10, and b = -10, our set of answers would be [-20, -10, 0, 10, 20, 30], whereas the correct answers are [-3, -1, 1, 3, 5, 7].

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

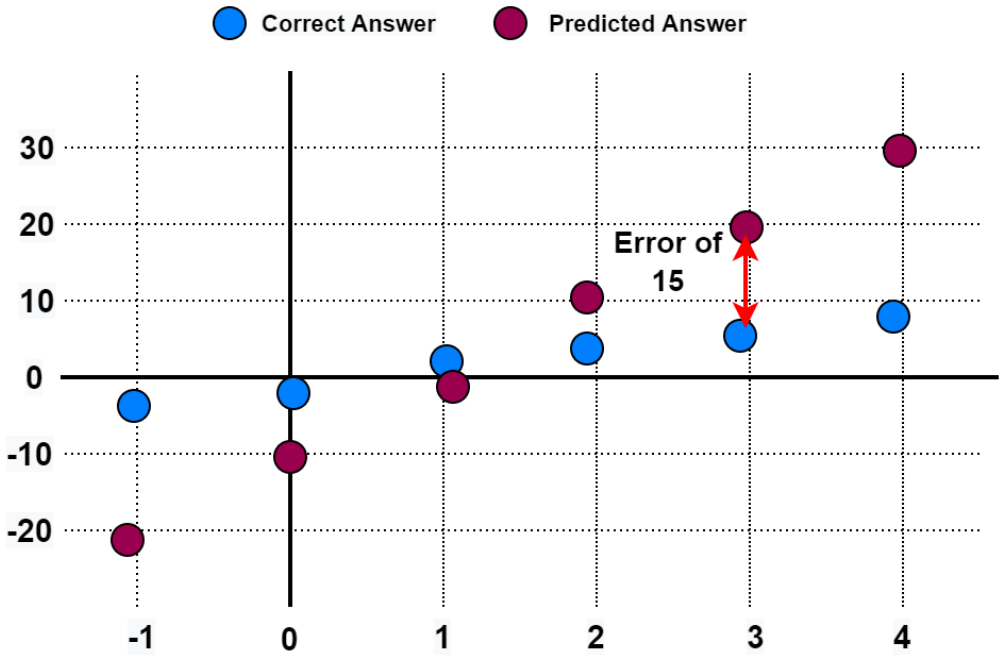
A scatter plot showing the correct answer vs. the predicted answer is plotted for the given guess. The correct answer dots start at y = -3 at x = -1 and increase linearly to y = 7 at x = 4. This is compared to predicted values of y = -20 at x = -1 increasing linearly to y = 30 at x = 4.



The same scatter plot from above is shown highlighting the error at x = -, showing an error of -17 (predicted value of -20 vs. actual value of -3).



Again, the same scatter plot from above is shown now highlighting the error at x = 3 of 15 (predicted value of 20 vs. actual value of 5).

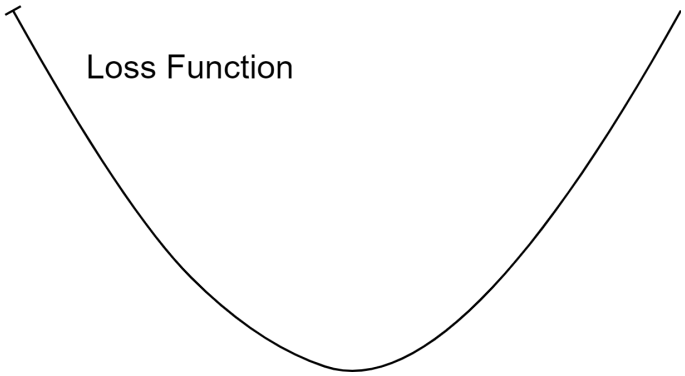


If we want to find out our overall error, we could average each of these.

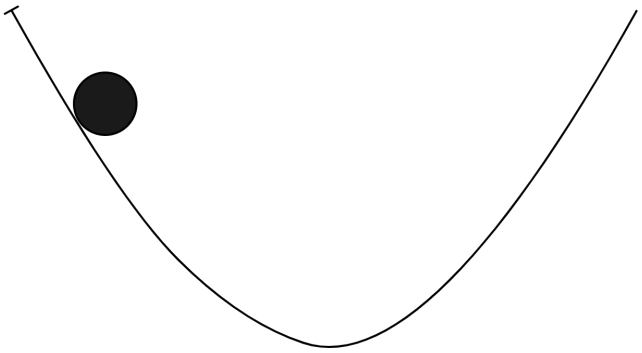
However, by doing this our negative (-17) and positive (15) errors would mostly cancel out. Therefore, the smart thing to do here is to square each error amount, average out all the individual squared errors, and then take the square root of that.

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

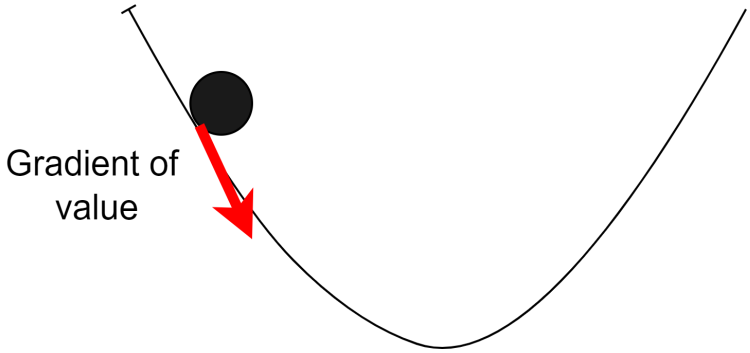
Now, when we want to optimize our guess, i.e. make it better than the previous guess, we can look at the loss function and figure out a way to minimize our loss. As the loss function involves the squaring of the error terms, the curve of the function is parabolic.



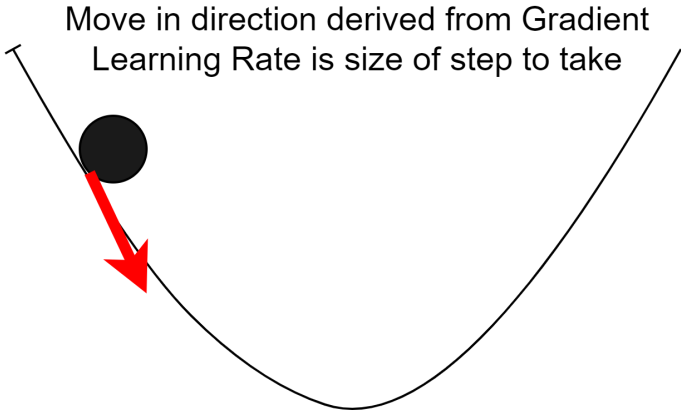
And the value of our loss with our current parameters can be plotted on this. Our loss was high, so we can say we're pretty far up the curve.



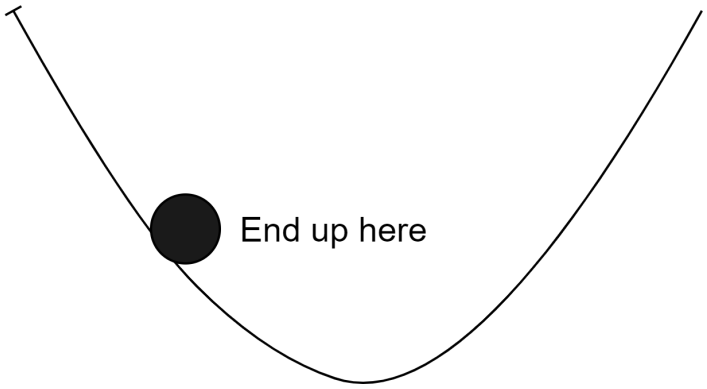
When we differentiate our values against the loss function, we'll get a gradient, and we can use the gradient to figure out which direction we have to go in to move down the slope!



And we can then move down the slope in the direction derived from the 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**.