***Neural Networks and Weights***

Using training data, we map (match) input values to output values using weights. This training data helps our network find patterns in the numbers to make more accurate predictions for future data.

So, how do Neural Networks figure out what weight to use?  
The answer is quite simple, Neural Networks start out with random numbers for the weights.

In the example below, our training data’s ideal weight is 2 (the previous year is doubled)

|  |  |
| --- | --- |
| Last Year Sales | This Year Sales |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |
| 8 | 16 |

With a weight of 2, when the new input (8) is entered  
 the Neural Network predicts the output will be (16).

Let’s try an example using the training data above.   
Our algorithm starts with a random value of **1.5** for weight.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Layer | Hidden Layer | Output Layer | Target | Error / Loss |
| 2 | W = 1.5 | 3 | 4 | 1 |

Since we know our data for this year was supposed to be 4, we can calculate how far we were off (our loss) using this function:

For our first example we have a loss of 1.

Let’s continue to our second example.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Layer | Hidden Layer | Output Layer | Target | Error / Loss |
| 2 | W = 1.5 | 3 | 4 | 1 |
| 3 | W = 1.5 | 4.5 | 6 | 1.5 |

Now we have finished with our training data it is time to calculate our total error/loss. It is as simple as adding all the results together.

The total error/loss of our first prediction is 2.5. The goal of our NN is to get this number as close to zero as possible.

Let’s update our weight to **1.75**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Layer | Hidden Layer | Output Layer | Target | Error / Loss |
| 2 | W = 1.75 | 3.5 | 4 | 1 |
| 3 | W = 1.75 | 5.25 | 6 | 1.5 |

*Total Loss* = 1.25

After our second time through our training data, it is clear our predictions are getting more accurate. Not only is the Output Layer getting closer, but the Total Loss is decreasing. This is all thanks to our ***weights***.

Weights are extremely important for neural networks during their training phase. The weights start random but are later adjusted using bias, activation functions, and algorithms for backpropagation / gradient descent. The examples shown don’t show any of these more advanced features of a NN. Just very base level description of how weights work.