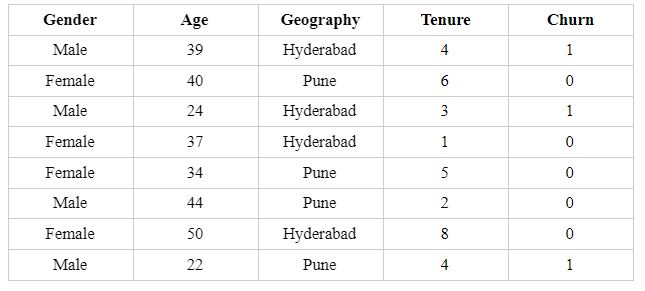
**Forward Propagation**

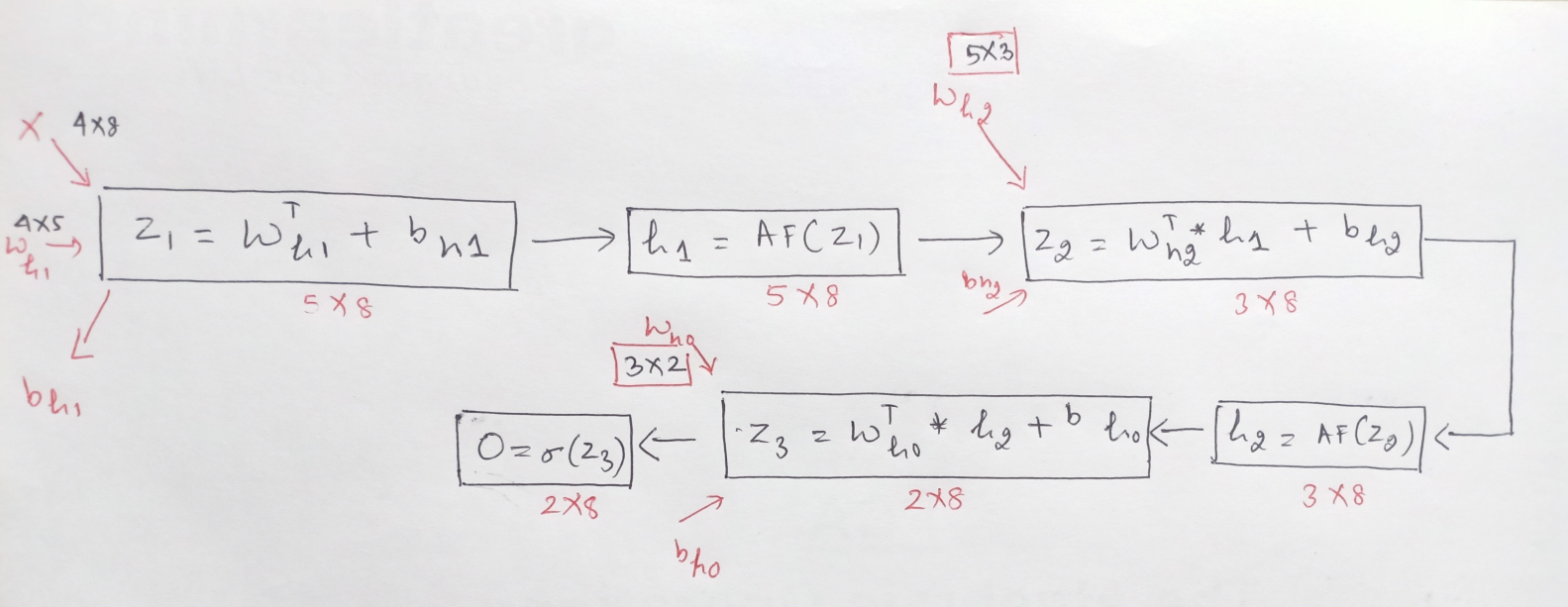
The process of going from **left to right** i.e from the **Input layer to the Output Layer** is **Forward Propagation**. We move from left to right to adjust or correct the weights. We will understand how this mathematically works and update the weights to have the minimized loss function.

Our binary classification dataset had input X as 4 \* 8 matrix with 4 input variables and 8 records and the Y variable is 2 \* 8 matrix with two columns, for class 1 and 0, respectively with 8 records. It had some categorical variables post converting it to dummy variables, we have the set as below:



The idea here is that we start with the input layer of the 4\*8 matrix and want to get the output of (2\*8). The hidden layers and the neurons in each of the hidden layers are hyperparameters and so are defined by the user. How we achieve the output is via matrix multiplication between the input variables and the weights of each layer.

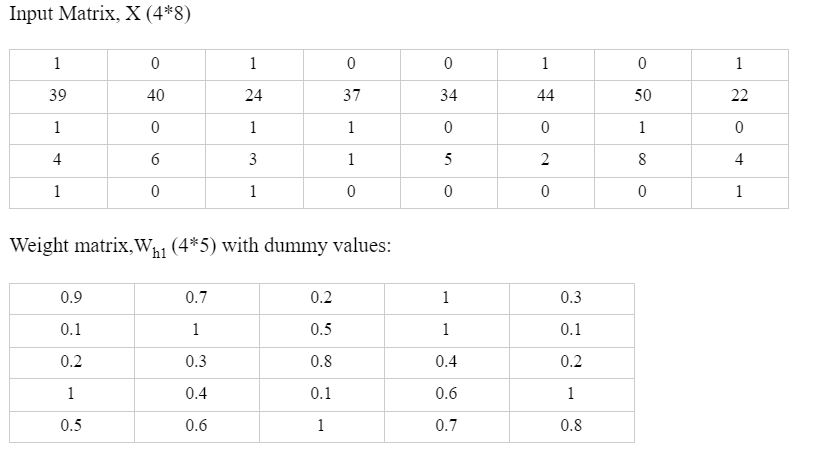
We have seen above that the weights will have a matrix for each of the respective layers. We perform matrix multiplication starting from the input matrix of 4 \* 8 with the weight matrix between the L1 and L2 layer to get the next matrix of the next layer L3. Similarly, we will do this for each layer and repeat the process until we reach the output layer of the dimensions 2 \* 8.



Please note the above explanation of the estimation of neurons was for a single observation, the network performs the entire above process for all the observations. The final result after performing for all the records in the dataset with the respective dimensions will be:

Now, let’s break down the steps to understand how the matrix multiplication in Forward propagation works:

1. First, the input matrix is 4 \* 8, and the weight matrix between L1 and L2, referring to it as Wh1 is 5 \* 5 (we saw this above).
2. The Wh1= 5\* 5 weight matrix, includes both for the betas or the coefficients and for the bias term.
3. For simplification, breaking the wh1 into beta weights and the bias (going forward will use this nomenclature). So the beta weights between L1 and L2 are of 4\*5 dimension (as have 4 input variables in L1 and 5 neurons in the Hidden Layer L2).
4. For understanding purpose, will illustrate the multiplication for one layer:



We can multiply element by element but that result will be only for one observation or one record. To get the result for all the 8 observations in one go, we need to multiply the two matrices.

For matrix multiplication, the number of columns of the first matrix must be equal to the number of rows of the second matrix. Our first matrix of input has 8 columns and the second matrix of weights has 4 rows hence, we can’t multiply the two matrices.

So, what do we do? We take the transpose of one of the matrices to conduct the multiplication. Transposing the weight matrix to 5 \* 4 will help us resolve this issue.

So, now after adding the bias term, the result between the input layer and the hidden layer L2, becomes Z1 = Wh1T \* X + bh1.

5. The next step is to apply the activation function on Z1. Note, the shape of Z1 does not change by applying the activation function so h1 = activation function(Z1) is of shape 5\*8.

6. In a similar manner to the above five steps, the network using the forward propagation gets the outcome of each layer:

Note that for the next layer between L2 and L3, the input this time will not be X but will be h1, which results from L1 and L2.

Z2 = Wh2T \* h1 + bh2,

where ,

* Wh2 is the weight matrix between L2 and L3 with a shape of 5\*3
* Wh2T , is the transpose of Wh2, having the dimension of 3\*5
* h1 is the result of L1 and L2, with a shape of 5\*8, and
* bh2 is the bias term.

So, Z2 = Wh2T \* h1 + bh2 with its matrix multiplication is:

Z2 = [(3\*5) \* (5\*8)] + bh2 will result Z2 with dimension of 3\*8 and post this again apply the activation function, which results in: h2 = activation function(Z2) is of shape 3\*8.

7. We repeat these steps for the computation of the last layer.

This time for the next layer between L3 and L4, the input will be h2, resulting from L2 and L3.

Z3 = Wh0T \* h2 + bh0,

* Where Wh0 is the weight matrix between L3 and L4 with a shape of 3\*2
* Wh0T, is the transpose of Wh0, having the dimension of 2\*3
* h2 is the result of L2 and L3, with a shape of 3\*8, and
* bh0 is the bias term.

So, Z3 = Wh0T \* h2 + bh0, with its matrix multiplication is:

Z3 = [(2\*3) \* (3\*8)] + bh0 will result in Z3 with the dimension of 2\*8 and post this again apply the activation function, this time use Sigmoid to transform as need to get the output, which results in O = Sigmoid(Z3) is of shape 2\*8.

After estimating the output using Forward propagation, then we calculate the error using this output, and this process of finding the weights to minimize the error continues until the optimal solution is achieved.

The other method and the preferred method to correct the weights is Backward Propagation which we shall explore in the class .