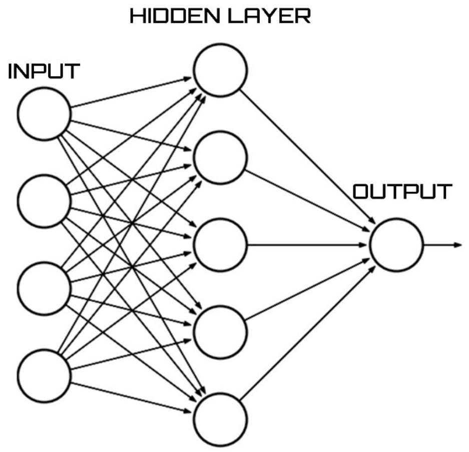
# THEORY ASSIGNMENT-1

## ● FORWARD AND BACKWARD PROPAGATION-

In neural networks the basic process involved is taking data input,manipulating the data with some mathematical parameters,calculating the deviation of our “mathematical output” with actual results in terms of error,and then update the parameters and repeat the entire process so as to minimize the error or in simple words try to match with actual results as much as we can.
So the process of calculating the mathematical output is called **forward propagation.** This process involves taking a weighted sum of inputs, applying the activation function…(these steps may be repeated )....finally calculating the output. And eventually the deviation of our output from actual results.
And then comes the other part “learning”. This involves updating the parameters in accordance with the deviation. And the process of calculating how much we want to change the parameters is called **backward propagation.**
Finally after repeating the process a few times we arrive at the parameters that help us to minimize the deviation. We use the same parameters to predict the output of new data.



## ● VECTORIZATION

In the above figure let's name the input layer elements as x[i], i is from 1 to 4. The outputs of the hidden layer are  named as a[j], j is from 1 to 5. And the final output is y[1].

❖ **Forward propagation steps**: this is all about calculating the deviation or the cost function,which will track our model's accuracy.

- The weighted sum for each hidden layer's unit can be calculated separately by using the parameters weights and bias as written below:

.

a[1] = w11*x [1] + w21*x[2] + w31*x[3] + w41*x[4]+ b1

▪

▪

▪

a[5] = w15*x [1] + w25*x[2] + w35*x[3] + w45*x[4]+ b5

The notation w-i-j stands for the i th weight of  j th hidden unit and b-i stands for the bias of i th hidden unit. Now the vectorization step just helps us to do the calculation in one step. And obviously to do so we must group everything we got….the inputs, outputs , weights ,biases separately.
We create following matrices(more generally speaking Tensors):

A(0) =

$$\begin{bmatrix} x[1] & x[2] & x[3] & x[4] \end{bmatrix}$$

W(1) =

$$\begin{bmatrix} w11 & w12 & w13 & w14 & w15 \\ w21 & w22 & w23 & w24 & w25 \\ w31 & w32 & w33 & w34 & w35 \\ w41 & w42 & w43 & w44 & w45 \end{bmatrix}$$

B(1) =

$$\begin{bmatrix} b1 & b2 & b3 & b4 & b5 \end{bmatrix}$$

A(1) =

$$\begin{bmatrix} a1 & a2 & a3 & a4 & a5 \end{bmatrix}$$

So writing the whole thing in mathematical form:

Z(1) = A(0) W(1) + B(1)

The next step is applying the activation function on the weighted outputs….so called activating step.let me store the final output in the same Tensor.

A(1) = g1(Z(1))................................( Tensor )

Some common functions “g1” used are ReLU,sigmoid,softmax,leaky ReLU,tanh etc.

We will do the same steps for the output layer, but as there is only one hidden layer the size of W(2) will be (5,1) and B(2) will be (1,1). The corresponding equation will be

Z(2) =A(1) W(2) + B(2)

A(2) =g2(Z(2) )

With the same conventions.Note : A(2) is (1,1) tensor, the output for our input.

- Now comes the final step of calculating the cost by using the cost function:

  $$J = \frac{1}{2 \cdot m} \sum_{1}^{m} L(A(2), Y)$$

  Here m represents the no of inputs. To be clear the whole thing A(0) was just one input with the no of features(just a name for individual units of our input) as 4.

  But here we are just focusing on one input so we need to calculate the loss by using the predefined function for loss L.for example : the NLLoss function

  $$L = -Y \log(A(2)) - (1 - Y) \log(1 - A(2))$$

  So that’s it-our forward propagation is completed.

## ❖ Back propagation steps:

In order to calculate by how much we need to change the parameters,we have to do following update step:

$$W_{updated} = W - \alpha \cdot \frac{\partial J}{\partial W}$$

$$B_{updated} = B - \alpha \cdot \frac{\partial J}{\partial B}$$

Here alpha denotes learning rate and J is the cost( which is half the average of loss over the whole dataset. As we have only one input so we will be calculating only the derivative of loss for one input. Showing the step for first hidden layer:

$$\frac{\partial L}{\partial Z(1)} = \frac{\partial L}{\partial Z(2)} W(2) * \frac{\partial g1(Z(1))}{\partial Z(1)}$$

$$\frac{\partial L}{\partial W(1)} = \frac{\partial L}{\partial Z(1)} A(0)$$

As clear from above steps the whole thing is about going backwards i.e. calculating the derivative first w.r.t. Z(2) and then w.r.t. Z(1).

Note : the derivatives are taken keeping in mind the tensor form. the “*” above means element wise multiplication.
Thus the back propagation step is also completed. All we need to do is to iterate over the data and the model will train itself.

- ## ACTIVATION FUNCTION

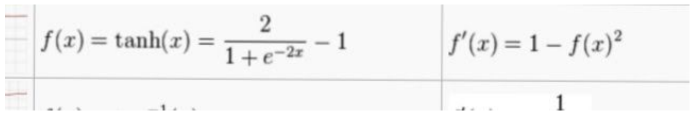
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = \sigma(x)\,(1-\sigma(x))$$

1. Sigmoid :
2. ReLU
   f(x)= x……..x>0
   f(x) =0……...x<0

   f'(x) = 1……..x>0
   f'(x)=0…….x<0
3. Leaky ReLU
   f(x)= x……..x>0
   f(x) =0.01x……...x<0

   f'(x) = 1……..x>0
   f'(x)=0.01…….x<0
4. tanh



| | |
|---|---|
| $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$ | $f'(x) = 1 - f(x)^2$ |

5.softmax

$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}}$$

Softmax function

where $\Theta = w_0x_0 + w_1x_1 + \ldots + w_kx_k = \sum_{i=0}^{k} w_ix_i = w^Tx$

$$Softmax = e^{x_a}/(\sum_{a=1}^{n} e^{x_a}) = e^{x_1}/(e^{x_1} + e^{x_2} + e^{x_3})$$

$$\frac{\partial(Softmax)}{\partial x_1} = (e^{x_1} \times (e^{x_2} + e^{x_3}))/(e^{x_1} + e^{x_2} + e^{x_3})^2$$

The softmax function's application includes other terms contribution so it is a little bit of complexity in the general formula so i avoided that.